Identificación de imágenes por análisis de texturas utilizando aprendizaje automático

Image identification using texture analysis and machine learning

Trabajo de fin de grado

Joaquín Barrio Lottmann
Xuebo Zhu Chen

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Curso académico 2019/2020

Identificación de imágenes por análisis de texturas utilizando aprendizaje automático

Image identification using texture analysis and machine learning

Memoria presentada para el trabajo de fin de grado de

Ingeniería Informática

Dirigida por el Doctor

José Jaime Ruz Ortiz

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Curso académico 2019/2020

Agradecimientos

Sin la ayuda incondicional de una serie de personas la consecución y finalización del proyecto no habría sido posible.

Para empezar, agradecer a nuestro tutor, José Jaime Ruz Ortiz, compartiendonos todo el material didáctico disponible para facilitar nuestro trabajo así como el conocimiento necesario para el desarrollo de la herramienta.

Dar las gracias también a nuestro círculo personal más cercano como familiares y amigos, gracias a su apoyo de manera directa o indirecta, por escucharnos y por dirigirnos en los momentos de mayor adversidad y duelo.

Agradecer también a todo el personal docente de la facultad por su excelente calidad como profesorado al enseñarnos y dotarnos de herramientas personales y profesionales para afrontar los retos laborales y académicos.

Finalmente agradecer a todos los autores y profesionales que elaboraron e investigaron sobre el temario en cuestión y por compartirlo públicamente para su difusión y consulta.

Resumen

Hoy en día, el tratamiento y reconocimiento de imágenes gracias a la inteligencia artificial y análisis de estas mediante aprendizaje automático nos reporta enormes beneficios y aplicaciones en el mundo moderno. A partir de estos mecanismos podemos pilotar vehículos de manera autónoma, reconocer expresiones o detectar enfermedades mediante el entrenamiento, refuerzo y test de la máquina.

Concretamente nuestro proyecto está dirigido al procesamiento digital mediante aprendizaje automático empleado para el análisis de texturas. A medida que avancemos en la lectura y desarrollo de esta memoria detallaremos sobre los métodos empleados, finalidad y objetivos del proyecto.

En términos generales, la finalidad última es desarrollar una máquina capaz de clasificar texturas a partir de una serie de imágenes de radiografías de la caja torácica que incluye mayormente los pulmones, con el motivo de detectar enfermedades respiratorias como la neumonía.

Para ello hemos dividido el proyecto en diversas fases debidamente diferenciadas:

- Investigación y trabajo previo.
- Implementación y desarrollo del programa.
    - Módulos 1, 2 y 3
- Pruebas y evaluación sobre imágenes torácicas de rayos-x con neumonía asociada al SARS-CoV-2.
- Síntesis y resultados finales.

Abstract

Today, the treatment and recognition of images thanks to artificial intelligence and the analysis of these through machine learning report the benefits and applications in the modern world. From these mechanisms we can pilot vehicles autonomously, recognize expressions or detect diseases through training, reinforcement and testing of the machine.

Specifically, our project is aimed at digital processing through machine learning used for the analysis of textures. As we continue reading and developing this report, we will detail the methods used, purposes and objectives of the project.

In general terms, the last one is a machine capable of classifying textures from a series of x-ray images of the rib cage that mainly includes the lungs, in order to detect respiratory diseases such as pneumonia.

To do this, we have divided the project into various different phases:

- Investigation and previous work.
- Implementation and development of the program.
  - Modules 1, 2 and 3
- Testing and evaluation with torax x-ray images of pneumonia associated to SARS-CoV-2.
- Synthesis and final results.

Palabras clave

- Análisis de textura
- Visión por computador
- Reconocimiento de patrones
- SVM
- Co-ocurrencia
- Python
- Análisis digital de imágenes
- Aprendizaje automático
- libSVM
- Dataset
- SARS-CoV-2

Keywords

- Texture analysis
- Computer vision
- Pattern recognition
- SVM
- Co-occurrence
- Python
- Digital image processing
- Machine Learning
- libSVM
- Dataset
- SARS-CoV-2

Índice

Agradecimientos V

Resumen VII

Abstract IX

Palabras clave XI

Keywords XIII

1. Introducción 2
1.1. Motivación 2
1.2. Objetivos 3
1.3. Visión general del documento 4
1.4. Motivation 5
1.5. Objectives 6
1.6. General vision of the document 6
2. Trabajo previo 8
2.1. Estado del arte 8
2.2. La matriz de co-ocurrencia 10
2.3. SVM 14
2.3.1. Definición del Hiperplano 15
2.4. SARS-CoV-2 16
2.5. Tecnologías relevantes 19
3. Metodología de trabajo 24
3.1. Google drive 24
3.2. GitKraken 25
3.3. GitLab 26
3.4. Skype 27
4. Identificación de imágenes por análisis de texturas usando aprendizaje automático 29
4.1. Funcionamiento general del sistema 29
4.2. Paquete Common 31
4.2.1 Componentes del paquete common 31
4.3. Módulo 1: Extracción de datos 32
4.3.1. Componentes del módulo 1 33
4.4. Módulo 2: Aprendizaje automático mediante SVM 33
4.4.1. Componentes del módulo 2 35
4.4. Módulo 3: Identificador de imágenes 36
4.4.1. Componentes del módulo 3 36

5. Pruebas y evaluación 39
5.1. Introducción 39
5.2. El problema 40
5.3. El conjunto de datos 40
5.4. El experimento 41
6. Conclusión y trabajo futuro 46
6.1. Conclusión 46
6.2. Conclusión 47
6.3. Trabajo futuro 47
7. Contribuciones individuales 52
7.1. Joaquín Barrio Lottmann 52
7.2. Xuebo Zhu Chen 53
8. Apéndice A Ejemplo de ejecución 55
9. Apéndice B Manual de usuario 62

Bibliografía 65

Índice de figuras

1.1 Correspondencia con las ocho direcciones cardinales 11
1.2 Definición y construcción de la matriz de co-ocurrencia 12
1.3 Ecuación de la Homogeneidad 12
1.4 Ecuación del contraste 12
1.5 Ecuación de la Disimilaridad 13
1.6 Ecuación de la GLCM Media 13
1.7 Ecuaciones de Varianza 13
1.8 Ecuación de la desviación estándar 13
1.9 Ecuación de la Entropía 13
1.10 Ecuación de la correlación 13
1.11 Ecuación del ASM 14
2.1 Dataset ejemplo SVM 14
2.2 Murciélago 17
2.3 Pangolín 18
2.4 Foco en China 18
2.5 Python 20
2.6 Ejemplo de primer uso Qt Designer 21
2.7 Creación de estructuras de datos con Numpy 22
3.1 Ejemplo de uso de la herramienta Google Drive 25
3.2 Ejemplo de uso de GitKraken 25
3.3 Ejemplo de uso de la aplicación web de GitLab 26
3.4 Ejemplo de uso de Skype para la comunicación grupal 27
4.1 Diagrama del sistema 30
4.2 Colocación de muestras de tamaño diverso en el módulo 1 32
4.3 Ejemplo GUI módulo 2 35
4.4 Ejemplo de uso del Módulo 3 37
5.1 Ejemplo de caso de prueba 3 43
5.2 Ejemplo de caso de prueba 7 44
6.1 Ejemplo de aplicación móvil detección enfermedades 49
6.2 Ejemplo de matriz de confusión para reconocimiento voz 50
A.1 Ejemplo de imagen virica 1 55
A.2 Ejemplo de imagen vírica 2 56
A.3 Ejemplo de imagen vírica 3 56
A.4 Ejemplo de imagen vírica analizada 57
A.5 Ejemplo de utilización de la herramienta 57
A.6 Ejemplo de uso de la herramienta módulo 2 58
A.7 Ejemplo de uso de la herramienta, ejecución exitosa 59
A.8 Ejemplo de uso de la herramienta (Módulo 3) 59
A.9 Ejemplo de uso de la herramienta (Imagen cargada) 60
A.10 Ejemplo de uso de la herramienta (Imagen procesada) 60

Capítulo 1

Introducción

RESUMEN: En esta introducción vamos a presentar la motivación y finalidad del proyecto realizado.

1.1. Motivación

La computación digital resuelve los problemas mediante procesos que emulan las capacidades del ser humano. A medida que la sociedad planteaba problemas más complejos se han ido incorporando soluciones cada vez más sofisticadas para dichos problemas.

Para un humano el reconocimiento de un objeto es sencillo, la clasificación de diferentes elementos no representa ningún problema, el problema parte cuando pretendemos identificar objetos desconocidos, o estamos todavía aprendiendo a identificarlos.

Dentro de la informática encontramos las técnicas de reconocimiento de imágenes por visión artificial, que permite el reconocimiento de objetos en imágenes. Puede ser útil para identificar ciertos objetos como podría ser el caso de ciertas patologías dentro del ámbito de la medicina.

En muchos casos un profesional es capaz de identificar sin problema una patología, pero un estudiante, en plena fase de aprendizaje le puede ser muy de gran ayuda el uso de una herramienta que facilite la retención del conocimiento.

Esto no solo es interesante para estudiantes sino también para la identificación de nuevas patologías pues los profesionales no disponen todavía de una amplia experiencia al ser casos previamente desconocidos.

En la actualidad encontramos diferentes enfoques para tratar el reconocimiento de objetos en imágenes, entre ellos podemos destacar:

- Aproximación basada en modelos.
- Métodos basados en apariencia.
- Métodos basados en características.

Dentro de los métodos basados en características podemos encontrar el análisis por texturas, que ha sido responsable de diferentes aplicaciones como la detección y diagnóstico del cáncer de piel y lesiones pigmentadas benignas (1991), diagnóstico de miocarditis (Ferdeghini, 1991) e identificación de personas mediante textura del iris.

Para abordar este problema es necesario un conjunto de herramientas que permitan realizar una clasificación mediante aprendizaje automático, nosotros utilizaremos una SVM que tomara los datos de un análisis digital previo.

1.2. Objetivos

El objetivo principal es el desarrollo de una aplicación cuya funcionalidad sea realizar un análisis basado en la textura que permita la detección y reconocimiento de objetos en imágenes. En este caso concreto la detección de un patrón en común en aquellas imágenes de Rayos X que confirmen la presencia de una neumonía asociada al SARS-CoV-2.

Primero habrá que realizar un análisis digital de las imágenes del dataset sobre el que se va a trabajar, de este se obtendrán muestras que usaremos más tarde para entrenar una SVM y clasificar imágenes con el resultado de este entrenamiento. Esto se conseguirá mediante un análisis de textura obteniendo sobre las muestras tomadas la matriz de co-ocurrencia y de 14 características calculadas sobre la matriz.

En segundo lugar, tendremos que declarar una SVM , que nos permitirá entrenar y clasificar. Como input recibirá los vectores de características de las muestras obtenidas en el primer módulo y las componentes propias de la matriz de co-ocurrencia.

Finalmente, mediante los datos obtenidos el los dos pasos anteriores, configuraremos la SVM para identificar los objetos muestreados inicialmente, pudiendo así obtener en un caso de éxito una respuesta fehaciente de si estamos ante una muestra con neumonía asociado SARS-CoV-2 o no.

1.3. Visión general del documento

La documentación de este trabajo se encuentra repartido en los siguientes capítulos:

- Capítulo 2: Expondremos una visión general del contexto y una investigación previa para el entendimiento y desarrollo del proyecto.
- Capítulo 3: Aquí detallaremos la coordinación llevada a cabo en el proyecto y las herramientas o utilidades usadas para el proyecto.
- Capítulo 4: Explicaremos la herramienta de forma atómica reduciéndose a cada una de sus módulos.
- Capítulo 5: Detallaremos el problema objetivo y realizaremos un estudio sobre el mismo.
- Capítulo 6: Conclusiones finales del proyecto y trabajo a futuro.
- Capítulo 7: Explicación de las contribuciones de los miembros del proyecto.
- Apéndice A: Ejemplo de ejecución de la aplicación. Apéndice B: Manual de usuario.

1.4. Motivation

Computation solves problems through processes that emulate the process capacity of the human being. As technology advanced, and society posed more complex problems, more sophisticated solutions to those problems were required.

For a human object recognition is a simple task, classifying different elements isn't a big problem, the trouble lies in identifying objects we're starting to learn about or classifying objects that are yet to be known for us.

Within the fields of computing we find the techniques of image recognition via artificial vision, this enables object detection in images which can be useful for people learning to identifying certain pathologies

In many cases a professional is capable of identificate pathologies without trouble, but for a student that is currently learning the experience is yet to be gained to just classify with a look, thus a tool that can classify would be useful in helping students learn.

Though it's not only interesting for students by also for new pathologies, identifying new pathologies might pose a problem for professionals too since the lack the experience as they haven't worked with many cases.

Nowadays we find different approaches to object detection the most notable are:

- Model based approximation methods.
- Appearance based methods.
- Feature based methods.

Within methods feature based there is texture based analysis, which has been responsible of different applications such as detection and diagnosis of skin cancer and pigmented skin injuries (1991), miocarditis diagnose (Ferdeghini, 1991) and facel identification via the iris texture.

To approach this problem there is need of a set of tools that allow us to make classification via machine learning, we will use a SVM that will input the data taken from a previous digital analysis.

1.5. Objectives

Our main objective is the development of an application that is able to make a texture based digital analysis that allows detection and recognition of objects in images. Specifically identifying SARS-CoV-2 associated pneumonia in x-ray images.

First, a digital analysis must be done with the image set, from it samples will be take to train the SVM and classify images with the result of that training. This will be accomplished by texture based analysis from the samples taking the co-occurrence matrix and 14 features taken from the matrix.

In second place, we must declare a SVM, that will allow us to train and classify with subsets of the image dataset. As input the SVM will take the feature vector from the samples taken and the data in the co-occurrence matrix.

Finally with the data obtain in the previous steps, the SVM will be configured to identify the sampled objects, thus being able to classify images that have SARS-CoV-2 associated pneumonia.

1.6. General vision of the document

The documentation of this work is distributed in the following chapters:

- Chapter 2: We will present an overview of the context and a previous investigation for the understanding and development of the project.
- Chapter 3: Here we will detail the coordination carried out in the project and the tools or utilities used for the project.
- Chapter 4: We will explain the tool in an atomic way reducing it to each one of its modules.
- Chapter 5: We will detail the objective problem and carry out a study on it.
- Chapter 6: Final conclusions of the project and future work.
- Chapter 7: Explanation of the contributions of the project members.
- Appendix A: Example of application execution. Appendix B: User Manual

Capítulo 2

Trabajo previo

RESUMEN: En el siguiente capítulo comentaremos el contexto en el que se sitúa nuestro proyecto y un estudio previo del mismo.

2.1. Estado del arte

Si nos metemos en contexto, en el ámbito de la visión por computador y el reconocimiento de objetos, nos derivamos al objetivo de cómo detectar la presencia de objetos o algún tipo de patrón en una secuencia de imágenes. Para ello hay que referenciar, por ejemplo, el software de detección de caras que se presentó con OpenCV.

OpenCV también conocido como Open Computer Vision, se ha usado en una gran cantidad de herramientas, conocida como el software de visión artificial más famoso del mundo. Con ello se logra una detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes entre otras aplicaciones.

Por lo[1] general en la detección y reconocimiento de objetos se pueden diferenciar dos partes, la extracción de características del contenido de una imagen y la búsqueda de objetos basada en dichas características. Respecto a la extracción de características, se fundamenta en la obtención de modelos matemáticos compactos que hagan una síntesis del contenido de la imagen con la finalidad de amenizar el proceso de aprendizaje de los objetos o patrones a reconocer.

Respecto al proceso de clasificación se pueden utilizar una variedad de técnicas de machine-learning. Algunos métodos como la regresión logística o en nuestro caso, en el que haremos especial hincapié, la técnica de aprendizaje automático conocido como el SVM.

Para poder hablar propiamente del reconocimientos de objetos, análisis de texturas, etc… hay que remontarse a los inicios de la visión artificial, y para ello hacer referencia a Larry Roberts, padre del término al crear y desarrollar un software que podía contemplar una estructura de bloques, analizar el contenido y replicar el objeto desde una perspectiva distinta, de esta forma él demostraba que la cantidad de información visual que había sido enviada al ordenador por una cámara, había sido procesada adecuadamente por él mismo.

[1] https://blog.infaimon.com/tipos-de-picking-en-el-almacen/

En adelante, aparecieron una serie de programas para aplicar esta solución al problema del reconocimiento y visión artificial en una gama de áreas distintas. Tales áreas objetivo como, detección, localización, segmentación y reconocimiento de objetos o patrones en imágenes como por ejemplo el reconocimiento de caras humanas. Posterior evaluación de los resultados, registro de diferentes imágenes de una escena u objeto, y hacer coincidir tal objeto con diversas imágenes, aplicaciones forenses y de investigación. También encontramos el seguimiento de un objeto en una secuencia de imágenes, mapeo de una escena para generar un modelo tridimensional de la misma dicho modelo podría ser usado por un robot para navegar por la escena, también la estimación de las posturas tridimensionales de humanos y la búsqueda de imágenes digitales por su contenido. Algo a tener en cuenta es que no se trata de algo meramente teórico sino que aquí se están resolviendo problemas de índole real.

En sí, tanto la evolución como las aplicaciones de la visión artificial han estado ligadas con el desarrollo de las cámaras fotográficas y obtención de imágenes desde la perspectiva científica que se produjo a partir del siglo XIX, como lo son las imágenes telescópicas, radiografías e imágenes tomadas a partir de Rayos X.

Tenemos que retrotraernos hasta la década de los 60 donde se dio el origen de un prototipo automatizado basado en cámaras de visión y sistema de procesamiento de las imágenes captadas. A partir de ello se intentaba encontrar determinadas estructuras y análisis del contenido, siempre con la ayuda del procesamiento de imágenes utilizando ordenadores.

El contexto y la historia de la visión artificial marca el hito en la década de los 80 con el desarrollo de la ingeniería informática y la creación de procesadores más sofisticados y rápidos, dando lugar a microprocesadores capaces de captar, procesar y reproducir imágenes tomadas por una cámara a la que podían estar conectada de forma remota.

A partir de aquí empezaron a explorar cómo los sistemas de visión artificial pueden capturar imágenes de forma automatizada y reproducir las características visuales de objetos o espacios a través de la interpretación de los datos que puede realizar un software destinado a esta labor.

Gracias a la visión artificial se han conseguido solucionar una serie de problemas actuales como, realizar el seguimiento de un objeto mediante la secuencia de imágenes, localizar y reconocer[2] ciertos objetos entre varios por sus características físicas, obtener modelos tridimensionales a través de los datos que genera el análisis de una imagen, estimar y comparar las medidas de varios objetos entre sí, crear modelos y patrones a través de archivos de imágenes digitales para realizar respuestas automatizadas.

Y estas actividades básicas de la visión artificial son las que se llevan perfeccionando desde hace décadas, ampliando sus aplicaciones en diferentes ámbitos y mejorando su uso en actividades tan variadas como los controles de calidad, o en ámbitos tan diversos, como el sector químico, alimentario o sanitario.

La llegada de una nueva era de lo que denominamos visión artificial amplía su potencial con *la Industria 4.0*[2] y el empleo de la alta tecnología en la automatización. En concreto, del empleo de la Inteligencia Artificial y la capacidad de un sistema automatizado para aprender y tomar decisiones según la información que él mismo capta y procesa.

El Machine Learning en conjunto con el Deep Learning están haciendo posible que la visión artificial opere como si se tratara de un ojo y cerebro humano por su capacidad para analizar, evaluar y tener en cuenta todas sus variables; pero con una mayor capacidad de respuesta por la posibilidad de procesar datos masivos, conocido como Big Data y analizarlos en tiempo récord, gracias a que estas tecnologías dotan de mayor capacidad de procesamiento e inteligencia a los sistemas automatizados.

2.2. La matriz de co-ocurrencia

El análisis de texturas se utiliza en varias aplicaciones, incluyendo la teledetección, la inspección automatizada y el procesamiento de imágenes médicas.

Se puede utilizar para encontrar los límites de textura, denominados segmentación de texturas, para esto se crea un modelo estadístico capaz de describir una textura mediante sus características. Este método puede ser útil cuando los objetos de una imagen se caracterizan más por su textura que por la intensidad, y las técnicas de umbral tradicionales no se pueden utilizar de forma eficaz.

La matriz de co-ocurrencia es base del modelo estadístico que describe la textura. Normalmente por matriz de co-ocurrencia entendemos como una estructura matemática que describe la frecuencia de los niveles de grises que aparecen en relación espacial específica con otros valores de grises dentro de una muestra de área específica. En resumen, se trata de la forma en que los valores de los pixeles ocurren al lado de otro valor en una pequeña ventana.

[2]https://blog.infaimon.com/ventajas-del-machine-learning-y-deep-learning-para-la-evolucion-de-la-vision-artificial/

La matriz de coocurrencia de niveles de gris[3] (GLCM) más los atributos derivados de ella son herramientas útiles para la clasificación de imágenes que detallaron inicialmente Haralick et al. (1973). La GLCM es una medida de frecuencia de diferentes combinaciones de valor de brillo de los píxeles en una imagen. Se utiliza ampliamente para clasificar imágenes de satélite (p.ej., Franklin et al., 2001; Tsai et al., 2007), imágenes de hielo marino (p.ej., Soh y Tsatsoulis, 1999; Maillard et al., 2005), imágenes de resonancia magnética y tomografía computarizada (p.ej., Kovalev et al., 2001; Zizzari et al., 2011) y en muchas otras aplicaciones. La mayoría de esas aplicaciones GLCM implican la clasificación de imágenes 2D.

Existen cinco variables fundamentales que en la generación de imágenes de textura: tamaño de la ventana, banda espectral de entrada, las texturas derivadas, cuantización del canal de salida y la componente espacial que vendría a ser la distancia interpixel y el ángulo para el cómputo de la co-ocurrencia.

Para el tamaño de dicha ventana su forma tiene que ser cuadrada. Asimismo, el resultado del cálculo de la textura es un único número que representa la ventana completa, este se coloca en el lugar del pixel central.

Con el análisis textural mediante las matrices de co-ocurrencia (GLCM) sobre imagen radiográfica de la caja torácica del paciente es de utilidad para la detección del patrón o objeto a reconocer en nuestro caso, una posible neumonía a causa del SARS-CoV-2. Para hacernos una idea, cuando las características basadas en estadísticos de primer orden no bastan, la extensión obvia es la computación de la matriz de co-ocurrencia para describir los estadísticos de segundo orden de dichas imágenes.

Para la posible definición de dicha matriz de co-ocurrencia, cada coeficiente es un número real, representado con un número discreto de niveles de grises.

Frecuentemente, la relación espacial entre el píxel de referencia y su vecino puede ser en cualesquiera de las ocho direcciones (Norte, Sur, Este, Oeste, Noreste, Sureste, Suroeste, Noroeste), una representación aproximada a nivel gráfico sería lo siguiente:

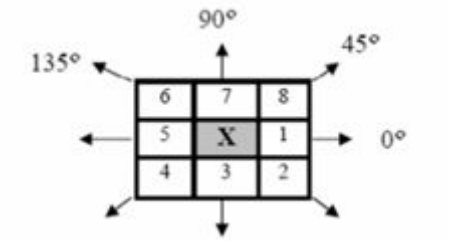

Figura 1.1: La correspondencia con las ocho direcciones cardinales para el cálculo de la matriz de co-ocurrencia.

[3] https://www.redalyc.org/pdf/1630/163013077004.pdf

Gracias a esta matriz se pueden calcular las 8 variables estadísticas de segundo orden que mencionamos anteriormente, propuestas por Haralick (1973). Se describirían propiedades como el contraste, la energía, entropía, entre otras.

A continuación, la siguiente figura muestra una reconstrucción de cómo se elaboraría detalladamente la matriz de co-ocurrencia en base a las muestras que le proporcionamos. Aquí se especifica a nivel general.

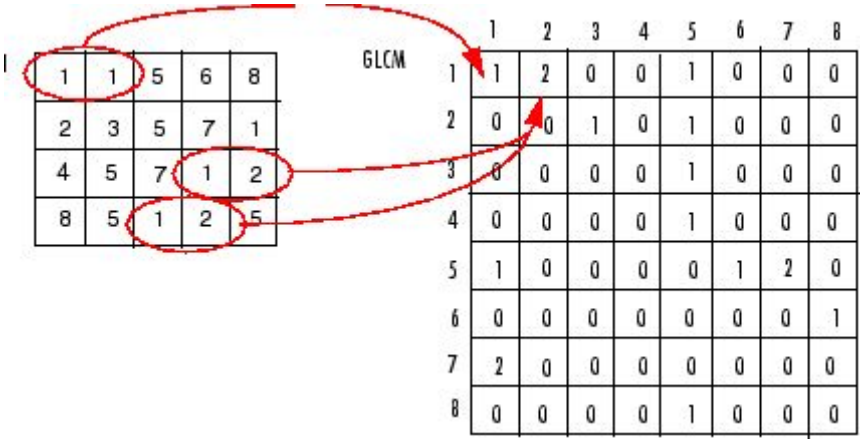

Figura 1.2: Definición y construcción de la matriz de co-ocurrencia.

En nuestro caso, utilizaremos las siguientes características, un rango de características que definió Haralick para el reconocimiento de patrones, cada una aportandonos información detallada y relevante de la textura en sí.

Por Homogeneidad entendemos el cálculo mediante una ecuación de la forma. Entendiendo P como la probabilidad de co-ocurrencia de los valores de gris i y j, para una distancia determinada.

$$\sum_{i,j=0}^{N-1} P_{i,j} / 1 + (i-j)^2$$

Figura 1.3: Ecuación de la Homogeneidad

Contraste, sería lo contrario a la homogeneidad, digamos una medida de variación local en una imagen.

$$\sum_{i,j=0}^{N-1} P_{i,j} (i-j)^2$$

Figura 1.4: Ecuación del Contraste

La Disimilaridad, similar al contraste, es elevada cuando la región tiene un contraste alto.

$$\sum_{i,j=0}^{N-1} P_{i,j}|i-j|$$

Figura 1.5: Ecuación de la Disimilaridad

GLCM Media, cálculo de la media aritmética de los valores de grises de los píxeles de la ventana.

$$\sum_{i,j=0}^{N-1} iP_{i,j}$$

Figura 1.6: Ecuación de la GLCM Media

Desviación Standard, se trata de la desviación estándar de los niveles de grises en la ventana. Valor elevado cuando la desviación estándar en los niveles de grises dentro de la ventana es también elevada. Los cálculos para la desviación estándar son las siguientes.

$$\sigma_i^2 = \sum_{i,j=0}^{N-1} P_{i,j}(i-\mu_i)^2 \qquad \sigma_j^2 = \sum_{i,j=0}^{N-1} P_{i,j}(i-\mu_j)^2$$

Figura 1.7: Ecuaciones de Varianza

Por otra parte las ecuaciones para el cálculo de la desviación estándar son las siguientes:

$$\sigma_i = \sqrt{\sigma_i^2} \quad \sigma_j = \sqrt{\sigma_j^2}$$

Figura 1.8: Ecuación de la desviación estándar.

Entropía, esta medida es alta en cuanto los elementos de la matriz de co-ocurrencia tienen relativamente valores similares.

$$\sum_{i,j}^{N-1} -P_{i,j}\ln(P_{i,j})$$

Figura 1.9: Ecuación de la Entropía

Correlación, proporciona una información bastante distinta con respecto a las otras medidas.

$$\sum_{i,j=0}^{N-1} P_{i,j}\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_i^2)}}\right]$$

Figura 1.10: Ecuación de la correlación

ASM (Angular Second Moment), en términos generales se trata de una medida de la homogeneidad local.

$$\sum_{i,j=0}^{N-1} P_{i,j}^2$$

Figura 1.11: Ecuación del ASM

2.3. SVM

Por SVM (Máquina de Vectores Soporte) entendemos un modelo de aprendizaje automático supervisado que utiliza algoritmos para problemas de clasificación de dos o más clases. Tras haberle dado a un modelo SVM un dataset de entrenamiento etiquetado para cada categoría, este es capaz de recategorizar.

Para entender cómo funciona una SVM, supongamos un ejemplo sencillo capaz de ilustrar su funcionamiento a nivel básico. Supongamos que en un plano de coordenadas tenemos una serie de valores de dos tipos, rojo y azul, dado un par de coordenadas nuevo queremos saber si este pertenece a la clase roja o azul. Visualmente el problema quedaría representado de la siguiente forma.

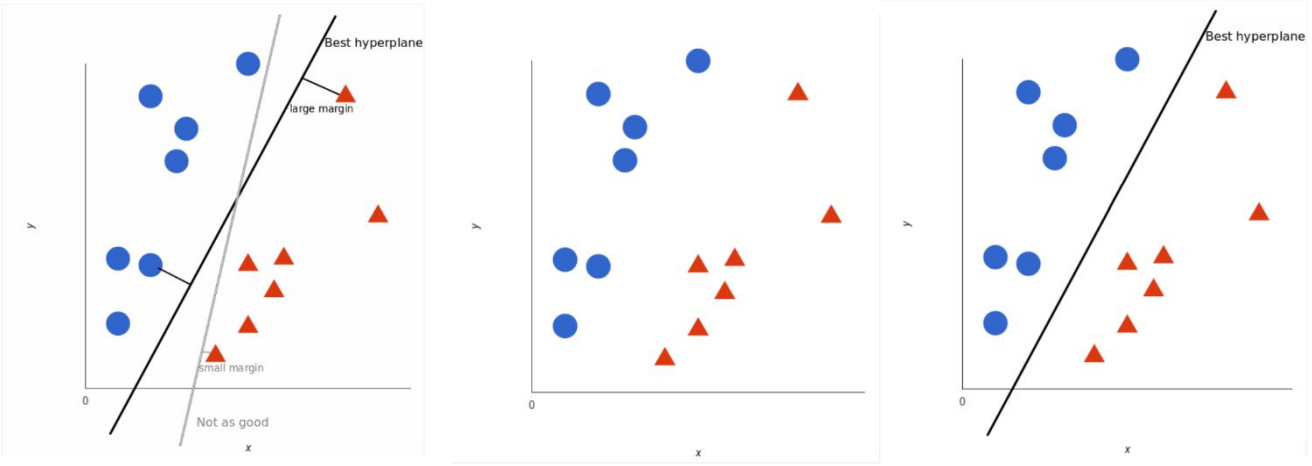

Figura 2.1: Dataset ejemplo SVM.

Una SVM recoge estos datos en el plano de coordenadas y realiza la separación de clases. La línea límite de decisión es la que remarca la frontera entre las dos clases. Con ello, la línea límite de decisión óptima es la que maximiza los márgenes desde el elemento más cercano de cada clase.

Las máquinas de vectores de soporte eran muy utilizadas antes de la era del aprendizaje profundo. Para muchas aplicaciones se prefería el uso de SVM en lugar de redes neuronales. La razón era que las matemáticas de los

SVM se entiende muy bien y la propiedad de obtener el margen de separación máximo era muy atractivo. Las redes neuronales podrían realizar clasificación de forma equivocada como hemos visto en los ejemplos anteriores.

Algunos casos de éxito de las máquinas de vectores de soporte son reconocimiento óptico de caracteres, detección de caras para que las cámaras digitales enfoquen correctamente, filtros de spam para correo electrónico, reconocimiento de imágenes a bordo de satélites (saber qué partes de una imagen tienen nubes, tierra, agua, hielo, etc.).

Actualmente, las redes neuronales profundas tienen una mayor capacidad de aprendizaje y generalización que los SVM.

En resumen, las Máquinas de Vectores de Soporte (Support Vector Machines) permiten encontrar la forma óptima de clasificar entre varias clases. La clasificación óptima se realiza maximizando el margen de separación entre las clases. Los vectores que definen el borde de esta separación son los vectores de soporte. En el caso de que las clases no sean linealmente separables, podemos usar el truco del kernel para añadir una dimensión

2.3.1. Definición del Hiperplano

En términos generales un *hiperplano* [4] es una extensión del concepto de plano. Si tratamos un espacio unidimensional, en este caso un hiperplano es un punto ya que divide esta en dos. Por otra parte si tratamos un espacio bidimensional, el hiperplano es una recta, ya que divide el plano en dos mitades. En un espacio tridimensional, un hiperplano es un plano al dividir el espacio en dos mitades. De este modo, se puede aplicar este concepto de manera sucesiva para más dimensiones.

Un hiperplano afín en un espacio n-dimensional puede ser descrito por una ecuación lineal no degenerada con la siguiente forma:

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = b$$

Aquí no degenerada significa que no todas las a_i son 0. Si b=0, se obtiene un hiperplano lineal, que pasa a través del origen.

Las dos mitades del espacio definidas por un hiperplano en espacios de n dimensiones son:

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \leq b$$

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \geq b$$

[4] https://es.wikipedia.org/wiki/Hiperplano

2.4. SARS-CoV-2

Los primeros análisis del SARS-CoV 2 indicaron que se trataba de un nuevo betacoronavirus que pertenecía a la misma familia del SARS-CoV y el MERS. Las investigaciones anteriores encontraron que el SARS-CoV se transmitió de gatos de civeta a humanos y el MERS-CoV de camellos y dromedarios a humanos. Asimismo, el SARS-CoV-2 es el séptimo coronavirus que se conoce que infecta a los humanos; el SARS-CoV, MERS-CoV y el SARS-CoV-2 pueden causar una enfermedad grave, mientras que HKU1, NL63, OC43 y 229E están asociados con síntomas leves.

Un estudio publicado en Nature Medicine repasa y compara todos los datos genómicos que se han analizado hasta ahora sobre el nuevo patógeno e hipotetiza sobre los escenarios en los que podría haber surgido: selección natural en un huésped animal antes de la transferencia zoonótica; y selección natural en humanos después de la transferencia zoonótica. Sintetizando, el SARS-CoV-2 no se deriva de ningún esqueleto de virus usado previamente.

2.4.1. Hipótesis del origen en laboratorio

Lo primero que puntualizan los investigadores es que descartan que se trate de un patógeno manipulado en el laboratorio. El RBD (el dominio de unión al receptor) de *SARS-CoV-2 está optimizado para unirse al ACE2 humano con una solución eficiente diferente a las comprobadas anteriormente*[5], argumentan en el trabajo, que ha sido elaborado por Kristian G. Andersen, del Departamento de Inmunología y Microbiología, Instituto de Investigación Scripps (EE.UU.) y sus colaboradores.

Además, si se hubiera realizado la manipulación genética, "se hubieran utilizado, probablemente, uno de los varios sistemas de genética inversa disponibles para los betacoronavirus", señala el equipo. Los datos genéticos muestran "irrefutablemente" que el SARS-CoV-2 no se deriva de ningún esqueleto de virus usado previamente. "Es probable que los murciélagos sirvan como reservorios para su progenitor".

[5] https://gacetamedica.com/investigacion/descifrando-los-origenes-del-sars-cov-2/

2.4.2. Enfermedad Zoonótica

Los primeros casos identificados en Wuhan (China) estaban vinculados a un mercado de animales, por lo que es posible que hubiese una fuente animal en dicho lugar.

Figura 2.2: Murciélago

Dada la similitud del SARS-CoV-2 con los coronavirus similares a los del SARS-CoV, es probable que los murciélagos sirvan como reservorios para su progenitor[4]

Sin embargo, aunque el virus RaTG13 de los murciélagos Rhinolophus affinis es idéntico al SARS-CoV-2 en un 96%, su pico diverge en el RBD, "lo que sugiere que puede no unirse de manera eficiente al ACE2 humano", explican en el estudio.

2.4.3. Huésped Intermedio

Los pangolines de Malasia importados ilegalmente a la provincia de Guangdong contienen coronavirus similares al SARS-CoV-2 . Aunque el virus RaTG13 sigue siendo el más cercano al SARS-CoV-2 en todo el genoma , algunos coronavirus pangolín exhiben una fuerte similitud con SARS-CoV-2 en el RBD. "Esto muestra claramente que la proteína de pico (S) de SARS-CoV-2 optimizada para unirse a ACE2 humano es el resultado de la selección natural", añaden los investigadores. "La diversidad de coronavirus en los murciélagos y otras especies se subestima masivamente".

A día de hoy no se ha identificado ningún coronavirus animal que sea lo suficientemente similar como para haber servido de progenitor directo del SARS-CoV-2, pero, tal y como señala el estudio, la diversidad de coronavirus en los murciélagos y otras especies se subestima "masivamente".

Figura 2.3: Pangolín

2.4.4. Selección natural en humanos

El otro escenario planteado sería selección natural en humanos después de la transferencia zoonótica. Este fenómeno explicaría que una versión no patógena del virus saltaría de un huésped animal a los humanos y evolucionaría a su estado patógeno actual dentro del sujeto.

El virus adquiere las características genómicas a través de la adaptación durante la transmisión no detectada de ser humano a ser humano. Una vez adquiridas, estas adaptaciones permiten que la pandemia empiece y produzca un grupo de casos suficientemente grande como para activar el sistema de vigilancia que lo detectó.

Figura 2.4: Foco en China

Los informes preliminares estimaron que el 'antepasado' común al SARS-CoV-2 apareció entre finales de noviembre de 2019 y principios de diciembre de 2019 , datos compatibles con los primeros casos confirmados retrospectivamente. "Por lo tanto, este escenario supone un período de

transmisión no reconocida en humanos entre el evento zoonótico inicial y su adaptación y replicación en humanos (adquisición del sitio de escisión polibásica)", según los datos del estudio.

Este escenario es el que dio lugar al MERS-CoV. Todos los casos humanos son el resultado de repetidos saltos del virus desde los camellos, produciendo infecciones únicas o cadenas de transmisión cortas que eventualmente se resuelven, sin adaptación a la transmisión sostenida.

2.4.5. Prevención a futuro de posibles eventos zoonóticos

Los investigadores concluyen que la comprensión detallada de cómo un virus animal saltó los límites de las especies para infectar a los humanos de manera productiva ayudará a prevenir futuros eventos zoonóticos. "Por ejemplo, si el SARS-CoV-2 se pre adapta en otra especie animal, existe el riesgo de futuros eventos de reaparición. Por el contrario, si el proceso de adaptación se produjo en humanos, incluso si se producen transferencias zoonóticas repetidas, es poco probable que despeguen sin la misma serie de mutaciones".

2.5. Tecnologías relevantes

2.5.1. Python

Python[6] es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C.
- La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.

[6] https://www.python.org/

Además, Python es gratuito, incluso para propósitos empresariales.

Figura 2.5: Logo de Python

El creador del lenguaje es un europeo llamado Guido Van Rossum. Hace ya más de una década que diseñó Python, ayudado y motivado por su experiencia en la creación de otro lenguaje llamado ABC. El objetivo de Guido era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando C.

El desarrollo de Python duró varios años, durante los que trabajó en diversas compañías de Estados Unidos. En el 2000 ya disponía de un producto bastante completo y un equipo de desarrollo con el que se había asociado incluso en proyectos empresariales. Actualmente trabaja en Zope, una plataforma de gestión de contenidos y servidor de aplicaciones para el web, por supuesto, programada por completo en Python

2.5.2. PyQt

PyQt[7] es un binding de la biblioteca gráfica Qt para el lenguaje de programación Python. La biblioteca está desarrollada por la firma británica Riverbank Computing y está disponible para Windows, GNU/Linux y Mac OS X bajo diferentes licencias.

[7] https://www.qt.io/

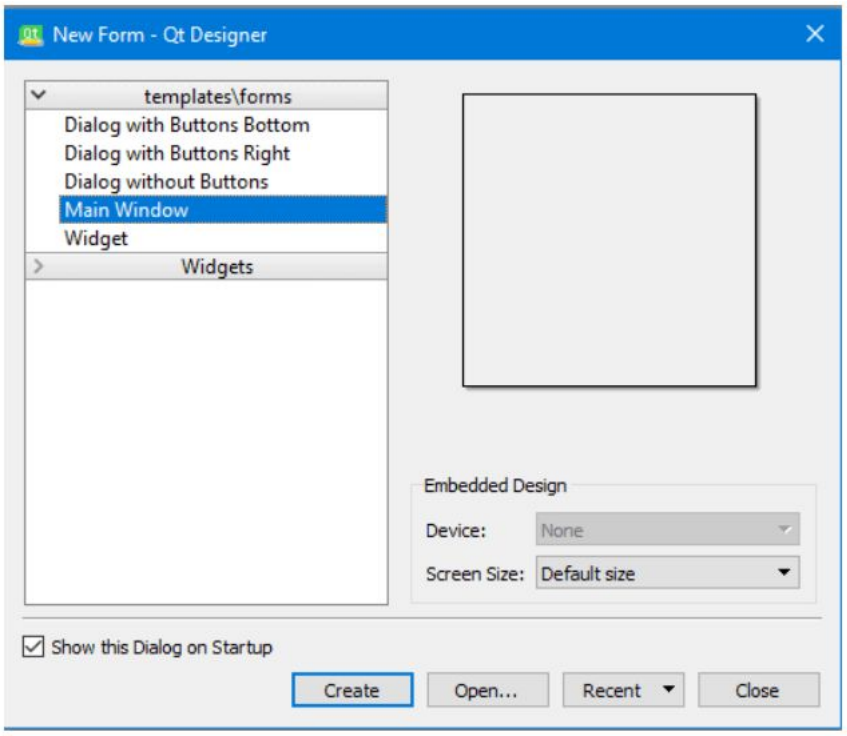

Figura 2.6: Ejemplo de primer uso Qt Designer

En agosto de 2009, tras intentar negociar con Riverbank Computing la liberación de PyQt bajo licencia LGPL sin conseguirlo, Nokia, propietaria de Qt, libera bajo esta licencia un binding similar, llamado PySide.

2.5.3. NumPy

NumPy[8] es, al igual que Pandas, Matplotlib o Scikit-Learn, uno de los paquetes que no puedes perderte cuando estás aprendiendo Machine Learning, principalmente porque esta librería proporciona una estructura de datos de matriz que tiene algunos beneficios sobre las listas regulares de Python. Algunos de estos beneficios son: ser más compacto, acceder más rápido a leer y escribir artículos, ser más conveniente y más eficiente.

NumPy es un paquete de Python que significa "Numerical Python", es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales. Estas estructuras de datos garantizan cálculos eficientes con matrices.

[8] https://numpy.org/

```
>>> a[0,3:5]
array([3,4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```

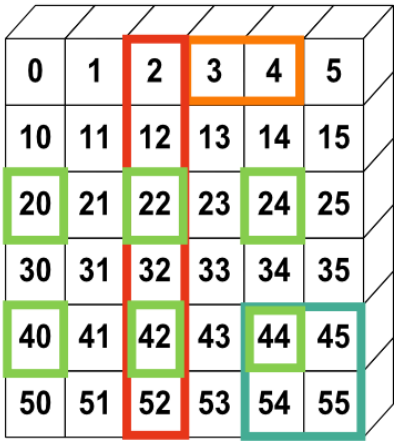

Figura 2.7: Creación de estructuras de datos con Numpy

2.5.4. LibSVM

LibSVM[9] es una de las librerías en código abierto más conocidas para trabajar con máquinas de vector soporte. Su licencia es BSD, por lo que hemos podido usar su código C para compilar nuestras propias versiones de los programas de entrenamiento y predicción.

Estas librerías contienen un amplio repertorio de APIs en diferentes lenguajes, por ejemplo en Java, C, C++, Python, R, etc. También trae soporte para apps de android.

Entre sus funcionalidades destacan los clasi cadores de máquina soporte (SVC), regresiones (SVR) y estimación de una distribución (SVM de una clase). Para nuestro trabajo nos enfocamos exclusivamente en los clasi cadores de máquina soporte, en especial C-SVC, que trabaja con el hiper-parámetro C, explicado anteriormente en la teoría.

Otra característica muy importante, es que contiene un script llamado grid.py, que se emplea para trabajar con validación cruzada. La validación cruzada es una técnica capaz de conseguir resultados óptimos de los valores C y gamma para un determinado caso de prueba con sus predictores.

En cuanto a kernels, libSVM soporta el kernel lineal, polinomial, radial (gaussiano) y el kernel sigmoide. En la parte práctica veremos que los kernels más útiles para nuestras clasi caciones serán el kernel polinomial y el radial.

[9] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Capítulo 3

Metodología de trabajo

RESUMEN: En dicho capítulo detallaremos la metodología de trabajo aplicada así como la organización del proyecto llevada a cabo. Uso de herramientas organizativas y seguimiento del control de versiones.

Para abordar la consecución del proyecto y su realización y control, primero teníamos que decidir qué herramientas utilizar para la organización y elaboración del mismo. Por tanto encontramos una serie de ramas e hitos del proyecto como investigación, desarrollo y trabajo a futuro.

A partir de la fase de investigación y recabado de información, la cual podemos dividir en lectura y comprensión de la bibliografía recomendada por el profesor respecto al temario del análisis de texturas y matriz de co-ocurrencia, pudimos segmentar la fase de desarrollo en una aplicación de 3 módulos.

Junto con la bibliografía recomendad del profesor más la información proporcionada por fuentes externas en Internet como Wikipedia o páginas oficiales completamos con éxito la fase de investigación y consecuentemente la fase de desarrollo.

3.1. Google drive

Desde un primer punto de vista no consideramos necesario ningún tipo de almacenamiento de datos en internet debido a que trabajabamos en el proyecto en persona, finalmente desde un punto de vista práctico vimos que una buena forma de archivar las herramientas y datos considerados de interés sería esta.

Google Drive es un servicio de almacenamiento cuya versión tal y como la conocemos hoy fue lanzada por Google en 2012.

Actualmente permite un almacenamiento gratuito de 15 GB para cada uno de sus usuarios y permite además visualizar y editar documentos entre varios usuarios a la vez, lo cual nos ha resultado muy útil de cara a la toma de apuntes a modo de bloc de notas.

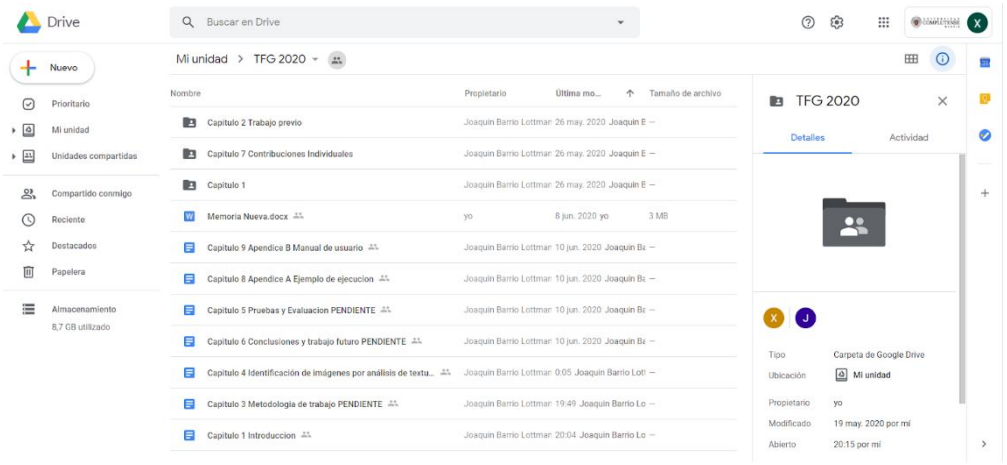

Figura 3.1: Ejemplo de uso de la herramienta Google Drive

3.2. GitKraken

Como herramienta para el control de versiones y actualización de repositorios del proyecto, hemos utilizado GitKraken. Herramienta de gran utilidad y renombre al ser de uso principal por empresas como Apple o Netflix. Con dicha aplicación encontramos una interfaz más amigable para el usuario con respecto a Git clásico.

Gracias a GitKraken se puede interactuar directamente con tu cuenta de Git sin tener que usar la línea de comandos por ejemplo. Decir que GitKraken se basa tanto en el uso de repositorios locales como de nube como podrían ser GitHub, GitLab y del TFS de Azure DevOps.

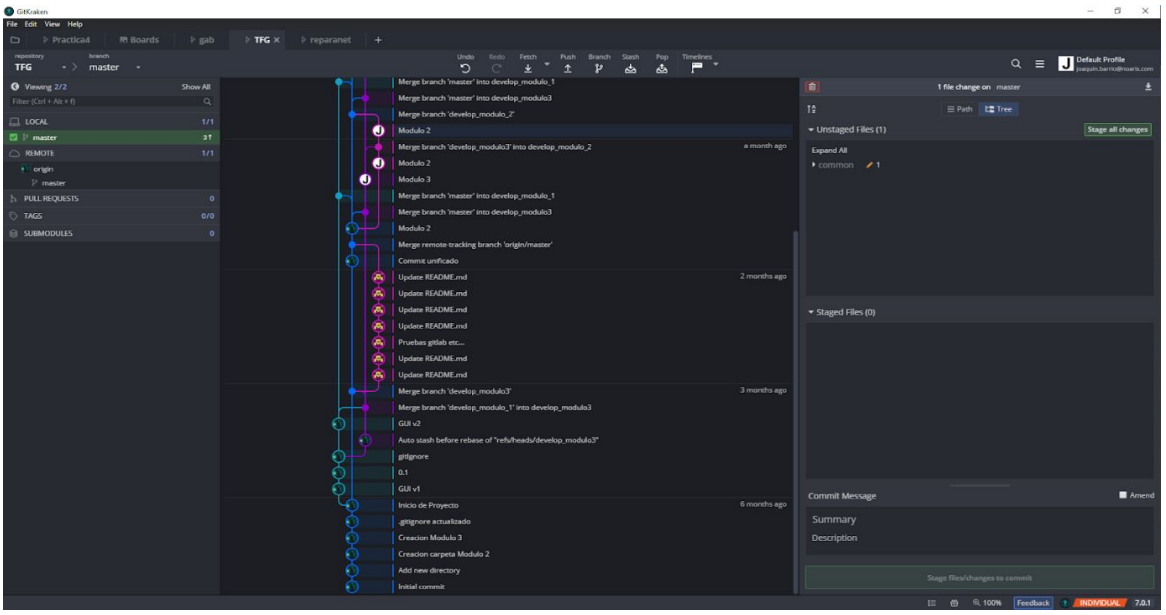

Figura 3.2: Ejemplo de uso de GitKraken

3.3. GitLab

Podemos decir que GitLab nació como un sistema de alojamiento de repositorios Git, es decir, un hosting para proyectos gestionados por el sistema de versiones Git. Sin embargo, alrededor de esta herramienta han surgido muchas otras herramientas muy interesantes para programadores y equipos de desarrollo, que envuelven todo el flujo del desarrollo y el despliegue de aplicaciones, test, etc.

Para dar una idea sintetizada lo más rápido sería compararlo con uno de sus competidores, GitHub, pues éste último es especialmente conocido en el mundo del desarrollo de software. Todos más o menos estamos familiarizados con lo que ofrece, la gestión de repositorios, las issues, los pull request, GitHub Pages, etc. GitLab sería algo muy similar a lo que encontramos en GitHub, aunque a veces con otros nombres. Otras alternativas de programas o servicios para Git como Bitbucket están muy por detrás en posibilidades.

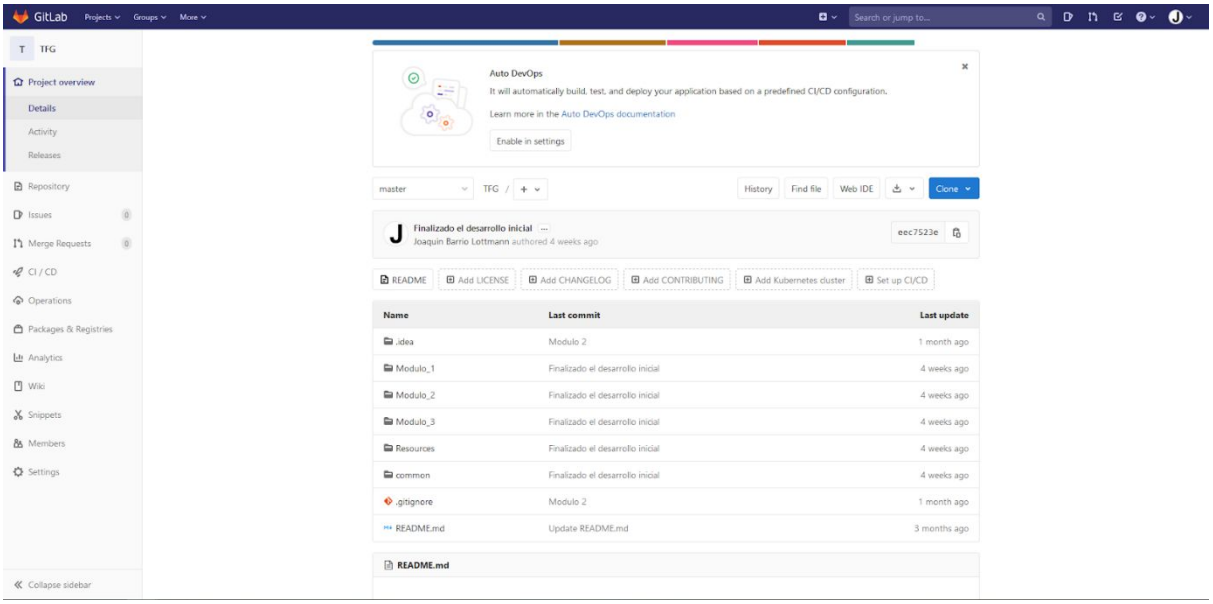

Figura 3.3: Ejemplo de uso de la aplicación web de GitLab

3.4. Skype

Respecto a la comunicación grupal y coordinación del equipo decidimos utilizar Skype como herramienta para realizar llamadas. Skype desarrollado principalmente por Microsoft. Nos permitió básicamente la comunicación gratuita por texto, voz o vídeo entre dos usuarios de Skype con computadores personales conectados a Internet. Requiere registro y aceptación de las condiciones del servicio, sin costo.

Skype al contar con los servicios de voz, datos, fax, contestador automático, conferencia y videollamada, puede mantener comunicación sin costo y a bajo costo, entre usuarios de Skype, teléfonos móviles, telefonos de red fija, fax, videoconferencias y obviamente texto, entre los usuarios en ambas direcciones comunicacionales.

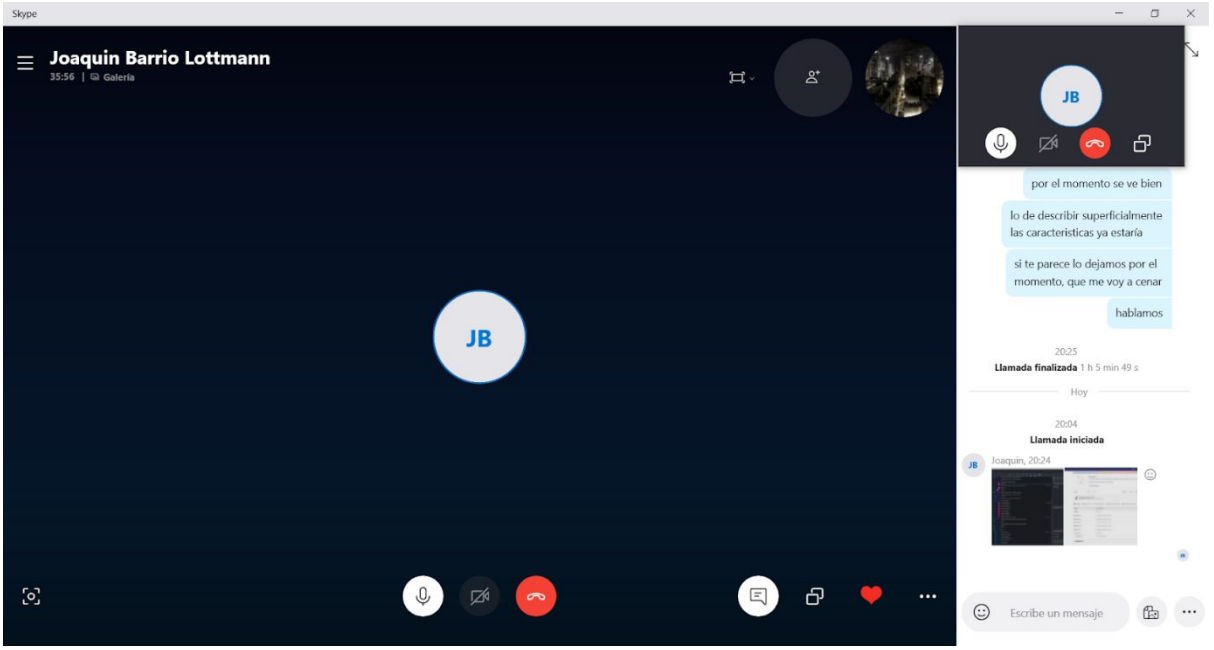

Figura 3.4: Ejemplo de uso de Skype para la comunicación grupal.

Capítulo 4

Sistema de identificación de imágenes por análisis de textura utilizando aprendizaje automático

RESUMEN: En este capítulo describiremos el sistema en su totalidad a partir de un análisis sobre cada una de las partes que lo componen.

4.1. Funcionamiento general del sistema

El proyecto está compuesto por diferentes módulos que se centran en diferentes áreas de estudio de datos que son parte de un análisis de imágenes por texturas.

En los diferentes módulos se trabaja con diferentes formatos de datos y estructuras sobre las características utilizadas en el estudio de forma que se puede obtener información de una imagen para poder clasificar otras identificando ciertos elementos a través de la textura.

El sistema está compuesto por tres módulos que se centran en tres diferentes partes del desarrollo del estudio de imágenes. El primer módulo permite analizar muestras, el segundo es un conjunto de herramientas de aprendizaje automático basados en una SVM para realizar el estudio sobre los datos obtenidos en el primer módulo, finalmente el tercer módulo del sistema realiza la identificación de imágenes.

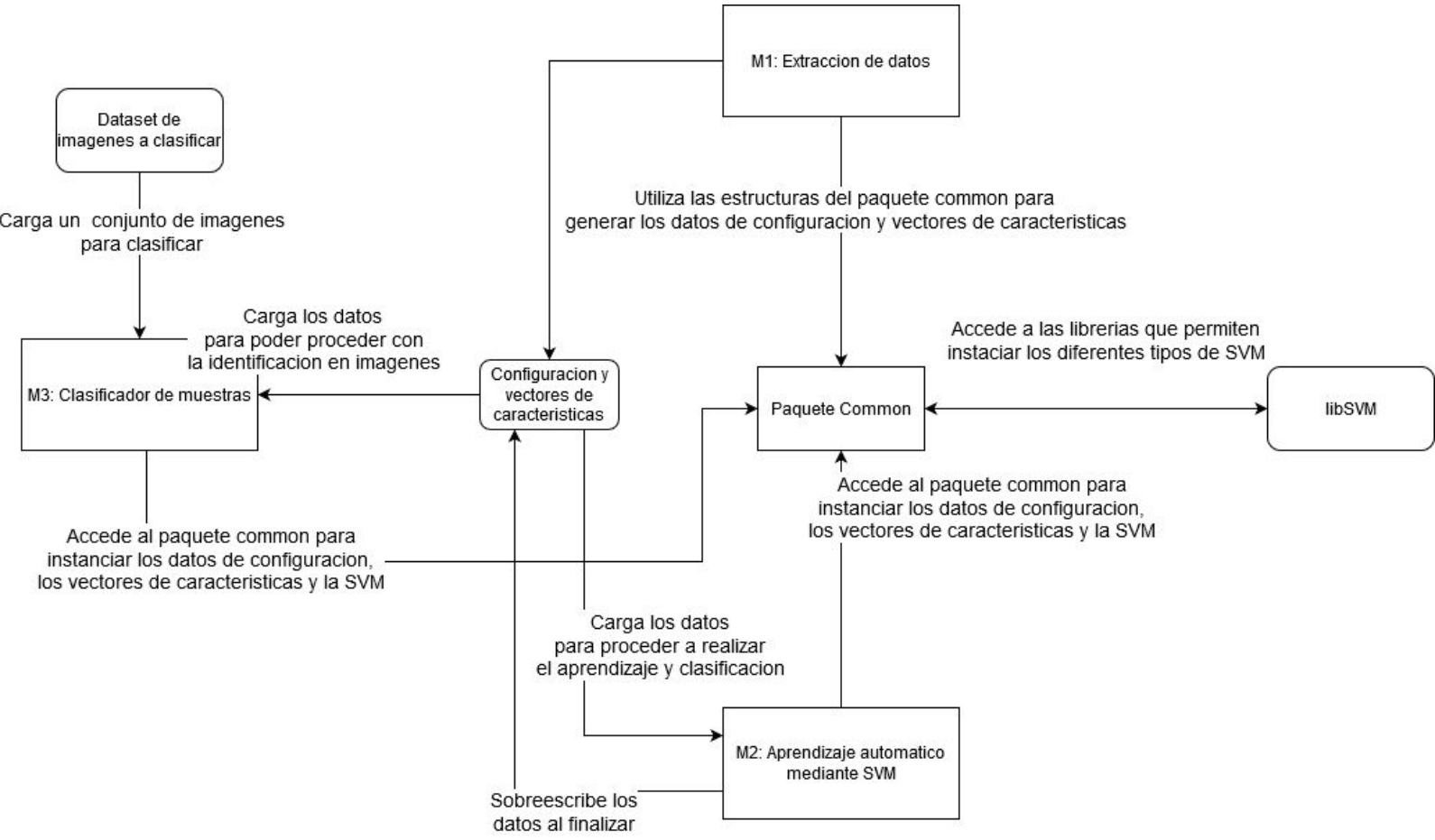

Figura 4.1: Diagrama del sistema

En la figura 4.1 se muestra el diagrama del sistema, compuesto principalmente por 3 módulos un paquete común y el DataSet sobre el que se realiza el estudio.

El paquete common contiene las estructuras de datos compartidas por los diferentes módulos y gestiona el acceso a libSVM la herramienta que utilizamos para instanciar SVMs.

El módulo 1 realizará la extracción de datos y análisis digital sobre muestras tomadas del DataSet.

El módulo 2 recibirá los datos del análisis como entrada y realizará el entrenamiento de la SVM

Finalmente en el módulo 3, con los datos obtenidos tras entrenar el SVM se encargará de realizar un barrido sobre una serie de imágenes para identificar los objetos clasificados.

4.2. Paquete Common

El paquete “common” es un módulo del sistema que contiene funciones, estructuras usadas y componentes usados por todo el sistema.

Está compuesto por:

- Estructuras de datos del dataset utilizado en los tres módulos para intercambiar la información entre los mismos.
- Herramientas de compresion y descompresion de archivos para poder contener toda la información del dataset en un único fichero.
- LibSVM componente que permite el uso de las diferentes sistemas de soporte de vectores.
- Funciones de cálculo para tomar las diferentes características de las muestras

4.2.1 Componentes del paquete common

- LibSVM: es un sistema integrado de software para clasificación mediante vectores de soporte
- PyQt: implementación para python del framework GUI Qt
- Numpy: paquete de python que contiene estructuras y soporte para trabajar con vectores y matrices.
- Scikit-image
- Pillow: paquete de python para el tratado de imágenes, contiene estructuras de datos y funciones que permiten trabajar sobre imágenes de manera sencilla.
- Paquete JSON de Python
- Paquete zipfile de Python

4.3. Módulo 1: Extracción de datos

El módulo uno realiza la extracción de datos, de las muestras del conjunto de imágenes. Las muestras serán seleccionadas sobre la imagen por el usuario y a partir del área seleccionada se extraerán datos según la configuración que haya preestablecido el usuario. Esta fase es probablemente la más importante pues el estudio se verá afectado en gran medida por qué datos se extraigan sobre las imágenes.

En la barra de herramientas que encontramos a la derecha podemos seleccionar una serie de opciones correspondientes al procesamiento de la muestras. Podemos indicar el tipo de clase (Positivo, Negativo), el tamaño, las características de la matriz de coocurrencia, es decir, la dirección de coocurrencia, la distancia, la escala de grises y por último los Estadísticos correspondientes a la matriz.

Una vez establecidas las características que se van analizar, tomando una imagen, colocaremos manualmente las muestras negativas y positivas, indicando en qué partes de la imagen el patrón de neumonía existe.

Tras haber definido todas las muestras que se van a tomar sobre una imagen o grupo de imágenes basta con procesarlas, y guardar el set de datos generado con la información de las mismas para proceder con el estudio.

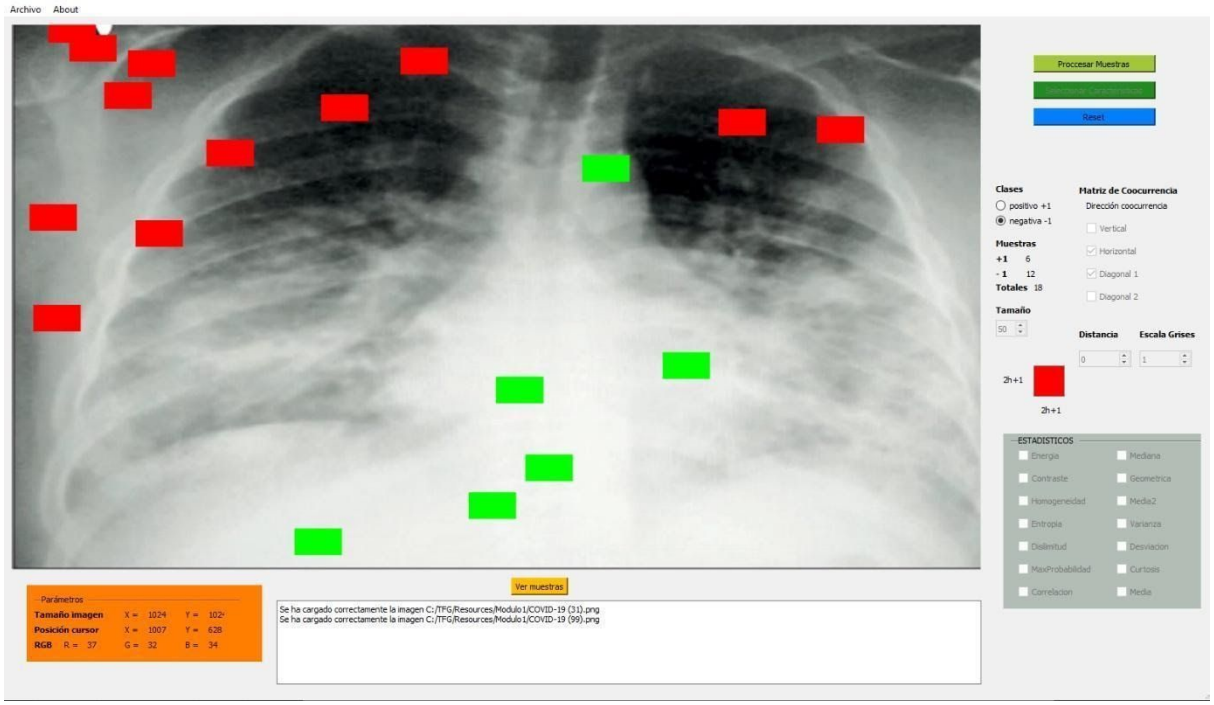

Figura 4.2: Colocación de muestras de tamaño diverso en el módulo 1.

4.3.1. Componentes del módulo 1

- Pillow: paquete de python para el tratado de imágenes, contiene estructuras de datos y funciones que permiten trabajar sobre imágenes de manera sencilla.
- PyQt: implementación para python del framework GUI Qt.
- Numpy: paquete de python que contiene estructuras y soporte para trabajar con vectores y matrices.
- Paquete threading de Python.
- Paquete Common

4.4. Módulo 2: Aprendizaje automático mediante SVM

El módulo dos se centra en el aprendizaje automático con los datos extraídos en el módulo anterior. Con esta herramienta el usuario toma los datos obtenidos en el primer módulo y aplicará un aprendizaje basado en el sistema y kernel seleccionado.

El aprendizaje se puede realizar con diferentes sistemas que pertenecen a libSVM la componente de este módulo que realiza el aprendizaje, estos sistemas que ofrece libSVM son los siguientes.

- SVM : este sistema trabaja sobre una única clase de datos de modo que ignorara las diferentes clases que le asignemos a nuestros datos, útil para analizar qué características son las mejore para ciertas imágenes antes de definir qué características usaremos para el estudio.
- c-SVC: este sistema emplea el parámetro denominado “coste” esta define el margen de la clasificación para el SVM.
- nu-SVC: este sistema el parámetro “nu” que indicará el número de vectores de soporte para cada clase indicada.
- Ɛ-SVR: SVM de regresión, este sistema emplea un, el parámetro denominado “epsilon”, este valor determina el nivel de precisión de la función de aproximación, el rango de este parámetro depende del set de datos con el que estamos trabajando.
- nu-SVR: SVM de regresión, este sistema emplea la variable “nu” del mismo modo que se daba anteriormente en el sistema nu-SVC.

Además del repertorio de sistemas anteriormente descritos libSVM nos permite hacer uso de los siguientes kernels

- Lineal
- Radial
- Sigmoide
- Polinómico

Finalmente, los parámetros aceptados (que dependen del sistema y kernel seleccionados) son los siguientes:

- Degree
- Gamma
- Coef0
- Cost
- Nu
- Épsilon-SVR
- Cache Size
- Epsilon
- Shrink
- Estimate
- Weight

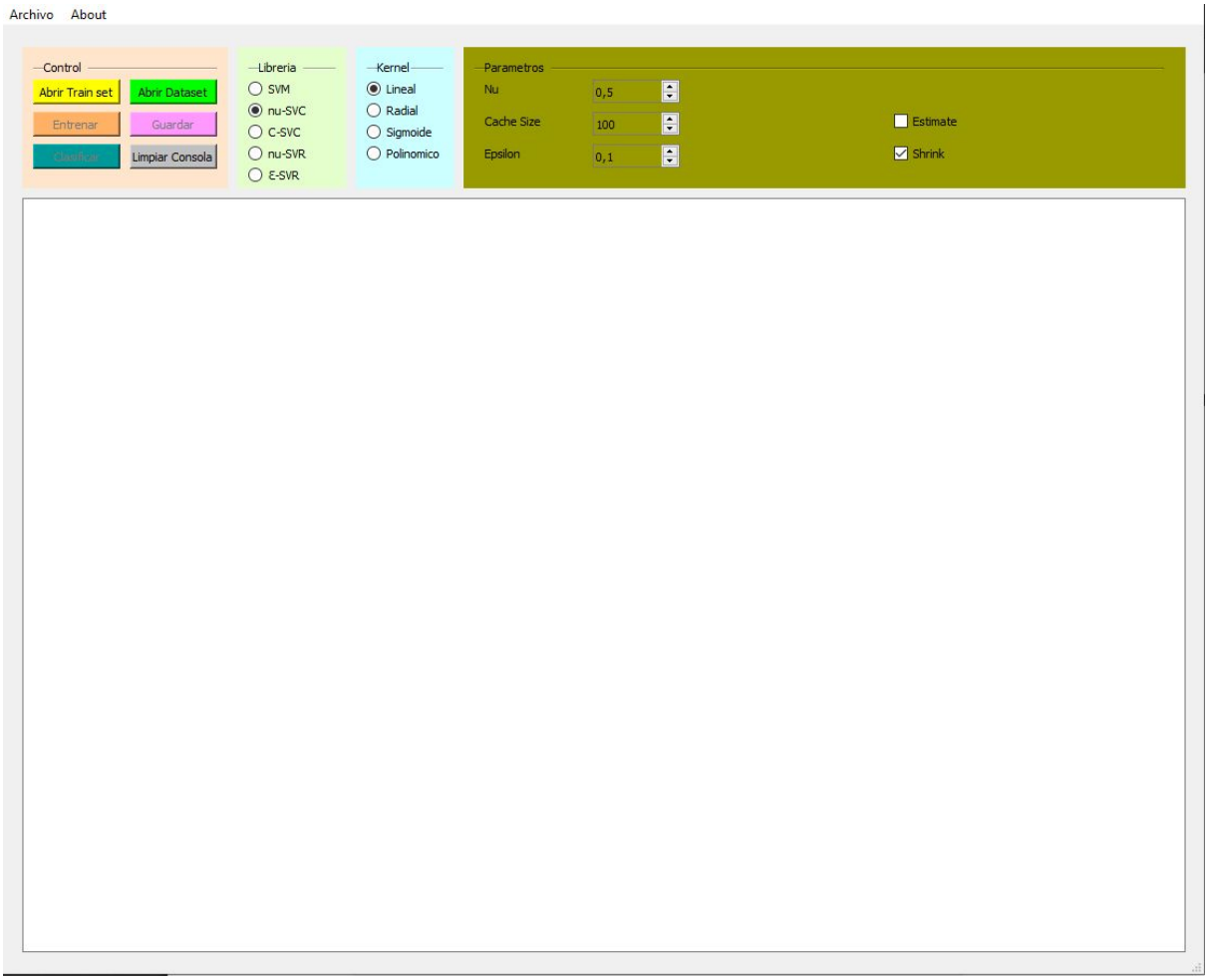

Figura 4.3: Ejemplo de la GUI del módulo 2. Distintas opciones de Librería y Kernels

4.4.1. Componentes del módulo 2

- PyQt: implementación para python del framework GUI Qt.
- Paquete threading de Python.
- Paquete Common

4.4. Módulo 3: Identificador de imágenes

El tercer módulo del sistema utiliza un modelo definido por el anterior módulo para identificar los elementos deseados en imágenes. Tras cargar un dataset toma la configuración y datos del mismo para configurar un sistema de libSVM, tras procesar una imagen con esta configuración "teñirá" la imagen en las áreas que no forman parte de la clase positiva, de esta forma el usuario fácilmente puede identificar las dos clases en la imagen mirando el resultado final.

Para empezar el sistema se cargará un dataset previamente definido mediante muestras tomadas en el primer módulo, y la configuración del segundo módulo para trabajar con libSVM, con esta información entrenará el sistema definido con los datos de las muestras, esto nos permitirá cargar imágenes para ser procesadas.

El procesado de la imagen consistirá en un barrido de la imagen tomando los datos de cada punto de la imagen dividiéndola en muestras semejantes a las tomadas en el primer módulo, esto se iniciará en diferentes procesos, tantos como núcleos lógicos disponga el sistema que está ejecutando la herramienta, el barrido generará una matriz de subimagenes, del tamaño de la muestra y cada proceso se encargará de procesar un grupo proporcional de las mismas.

Procesar las subimagenes consistirá en tomar los datos que concuerdan con los definidos en el primer módulo y clasificar la muestra con el modelo definido al cargar el dataset, según el resultado obtenido se marcará como clase positiva o clase negativa.

Una vez finalizado todos los procesos que están clasificando los diferentes fragmentos de la imagen se tomarán los resultados tiñendo la imagen de rojo en aquellos puntos que han sido clasificados como clase negativa.

4.3.1. Componentes del módulo 3

- Pillow: paquete de python para el tratado de imágenes, contiene estructuras de datos y funciones que permiten trabajar sobre imágenes de manera sencilla.
- PyQt: implementación para python del framework GUI Qt.
- Numpy: paquete de python que contiene estructuras y soporte para trabajar con vectores y matrices.
- Paquete zipfile de Python.
- Paquete threading de Python.

- Paquete multiprocessing de Python.
- Paquete Common

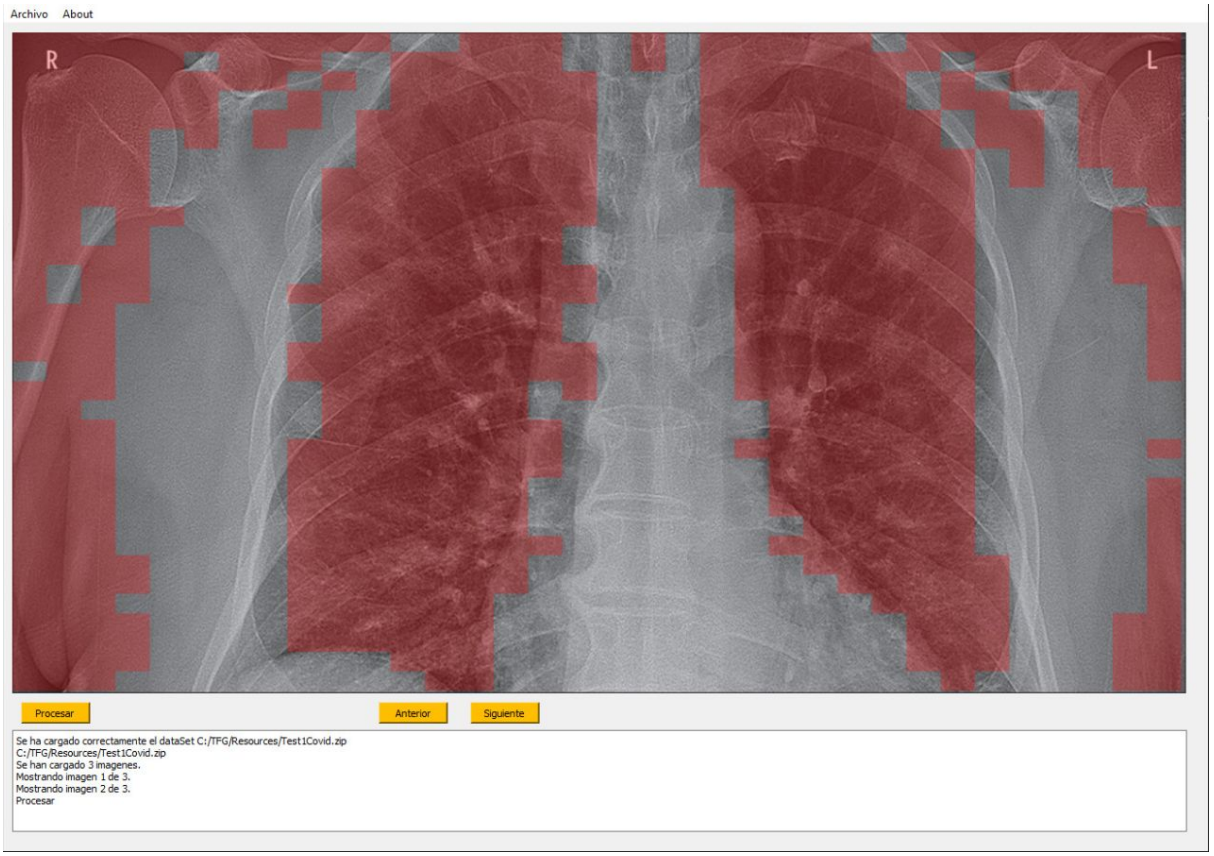

Figura 4.4: Ejemplo de uso del Módulo 3.

Capítulo 5

Pruebas y evaluación

RESUMEN: En este capítulo describiremos el sistema en su totalidad a partir de un análisis sobre cada una de las partes que lo componen.

5.1. Introducción

Al comenzar a trabajar en el proyecto no sabíamos sobre que realizar el análisis, ya que desconocemos que puede ser más susceptible o no, de cara a tomar datos mediante un análisis por texturas.

Inicialmente cuando surgió el problema del virus SARS-CoV-2, nuestro planteamiento fue aplicar las herramientas de análisis en las que estábamos trabajando para analizar muestras del virus, creíamos que esto podría ser útil de cara a identificar a infectados tomando muestras, pasadas por laboratorio, o incluso de cara a la investigación sobre el virus mismo.

Tras un tiempo sin encontrar un tema concreto, o para el caso un set de imágenes sobre las que trabajar, nuestro tutor nos pasó un dataset de imágenes de pacientes infectados con SARS-CoV-2 que habían desarrollado neumonía, esto nos pareció interesante pues se podía trabajar en clasificar la neumonía cuando viene acompañada del SARS-CoV-2, de esta forma en caso de tener algún éxito se podrían utilizar radiografías de pacientes para revisar si la enfermedad respiratoria de la que padecen esta asociada al virus.

El problema que tuvimos inicialmente con cierto dataset venía dado por la cantidad de imágenes que traía, si es verdad, que con el tiempo han ido avanzando, pero en su momento nos pareció que era demasiado pequeño el set y buscábamos al menos poder realizar pruebas en diferentes imágenes, garantizando que no realizamos un sesgo probando siempre sobre las mismas imágenes de cara a la toma de muestras, ya que quizás podríamos haber derivado en clasificar X imágenes con alto éxito, pero que dichas muestras no fueran efectivas de cara a imágenes fuera de esas iniciales X.

Por otro lado, nos topamos con un dataset de kaggle que contenía bastantes imágenes y disponía además de una gran cantidad de radiografías de pacientes que no padecen nada, o que padecían una neumonía viral, pero no asociada al SARS-CoV-2, con esto podríamos desarrollar el estudio y ver hasta qué punto nuestras configuraciones y toma de muestras podían identificar el impacto del virus o si solo identifican la neumonía sin más.

5.2. El problema

Anteriormente se ha hablado del SARS-CoV-2 por lo que aquí se expandirá algo más en cuanto a la neumonía, que no ha sido mencionada anteriormente.

Esta es una enfermedad del sistema respiratorio que puede ser causada por hongos, virus o bacterias, es un tipo de infección aguda sobre los pulmones, concretamente los alvéolos de los pulmones, estos -en las personas sanas- se llenan del aire que respiramos, en los pacientes que padecen neumonía los alvéolos se encuentran llenos de pus y líquido, resultado en una respiración dolorosa que además impide un funcionamiento correcto de modo que se limita el oxígeno obtenido al tomar aire.

La neumonía se considera la principal causa de muerte infantil en todo el mundo, y las estadísticas la sitúan en torno a un 15% de las defunciones de niños menores de 5 años en todo el mundo. Las localizaciones con mayor prevalencia son el África subsahariana y Asia meridional.

Por otro lado las estadísticas sobre la neumonía asociada al SARS-CoV-2 implican en torno a un 50% de las muertes registradas en pacientes identificados con SARS-CoV-2.

5.3. El conjunto de datos

El conjunto de datos proviene de un grupo de investigación de las siguientes universidades Qatar University, Doha, la universidad de Dhaka y Bangladesh junto a algunos colaboradores malasios y pakistaníes que han colaborado para crear este conjunto de imágenes, el dataset está nombrado como *COVID-19 RADIOGRAPHY DATABASE*[10].

El conjunto de imágenes está compuesto por radiografías de tórax, en formato PNG con resolución de 1024 x 1024, las imágenes en su gran mayoría están tomadas frontalmente a excepción de algunos casos donde se toman de lateral, el set está compuesto por 3 carpetas que dividen las imágenes por clasificación, una con 219 radiografías con neumonía asociada a SARS-CoV-2, otra con 1341 imágenes de pacientes sin ninguna infección, y finalmente una carpeta con 1345 radiografías con neumonía vírica no asociada al SARS-CoV-2.

[10] Créditos por el conjunto de datos a, M.E.H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M.A. Kadir, Z.B. Mahbub, K.R. Islam, M.S. Khan, A. Iqbal, N. Al-Emadi, M.B.I. Reaz, "Can AI help in screening Viral and COVID-19 pneumonia?" arXiv preprint, 29 March 2020

5.4. El experimento

El experimento no ha dado resultados demasiado conclusivos, en algunos casos se ha conseguido clasificar la neumonía, pero nunca que esta venga exclusivamente asociada al SARS-CoV-2 o por lo menos identifique en bajo porcentaje la neumonia virica de una manera aceptable.

Casos de prueba, para todos se ha acabado usando la SVM c-SVC debide a que el resto de sistemas arrojaban resultados con precisión muy baja, lo cual nos puede hacer entender que los sistemas con regresión no son los mejores para este tipo de problema.

- Caso 1 :
    - Tamaño de muestra: 30
    - Escala de grises: 32
    - Características: Todas las características del sistema
    - Kernel: Lineal
    - Muestras tomadas: 100 muestras tomadas
    - Ángulos de la matriz de co-ocurrencia: 0, $\pi/4$, $\pi/2$, $3\pi/4$
- Caso 2 :
    - Tamaño de muestra: 50
    - Escala de grises: 32
    - Características: correlación, mediana, geométrica, varianza, desviación, media
    - Kernel: lineal
    - Muestras tomadas: 162
    - Ángulos de la matriz de co-ocurrencia: 0, $\pi/4$, $\pi/2$, $3\pi/4$
- Caso 3 :
    - Tamaño de muestra: 30
    - Escala de grises: 32
    - Características: todas salvo la varianza

- Kernel: lineal
- Muestras tomadas: 440
- Ángulos de la matriz de co-ocurrencia: 0, $\pi/4$, $\pi/2$, $3\pi/4$

- Caso 4 :
  - Tamaño de muestra: 20
  - Escala de grises: 32
  - Características: no se calcula ninguna característica se toma solo la matriz de co-ocurrencia
  - Kernel: lineal
  - Muestras tomadas: 470
  - Ángulos de la matriz de co-ocurrencia: 0, $\pi/2$
- Caso 5 :
  - Tamaño de muestra: 30
  - Escala de grises: 16
  - Características: Energía, contraste, homogeneidad, entropía, disimilitud, Max probabilidad, correlación, mediana y geométrica.
  - Kernel: lineal
  - Muestras tomadas: 380
  - Ángulos de la matriz de co-ocurrencia: 0, $\pi/4$, $\pi/2$, $3\pi/4$
- Caso 6 :
  - Tamaño de muestra: 30
  - Escala de grises: 16
  - Características: Energía, contraste, homogeneidad, entropía, disimilitud, max probabilidad, correlación, mediana y geométrica.
  - Kernel: lineal
  - Muestras tomadas: 380
  - Ángulos de la matriz de co-ocurrencia: 0, $3\pi/4$

- Caso 7:
    - Tamaño de muestra: 30
    - Escala de grises: 16
    - Características: Todas las características del sistema
    - Kernel: lineal
    - Muestras tomadas: 660
    - Ángulos de la matriz de co-ocurrencia: $0,\ \pi/4\ ,\ \pi/2\ ,\ 3\pi/4$

Otros problemas que han surgido con el estudio se dan en torno a la los elementos que aparecen en las radiografías más allá de la neumonía, como pueden ser los pulmones de los pacientes y los esqueletos de los mismos.

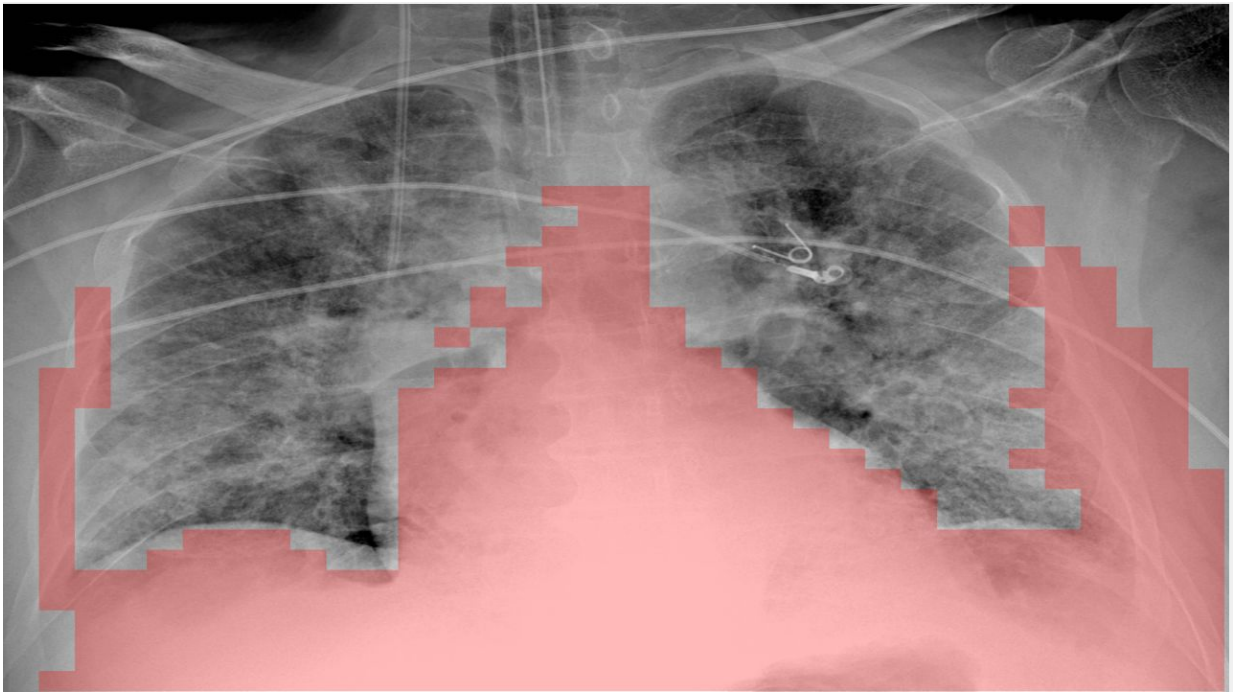

Figura 5.1: Ejemplo de caso de prueba 3

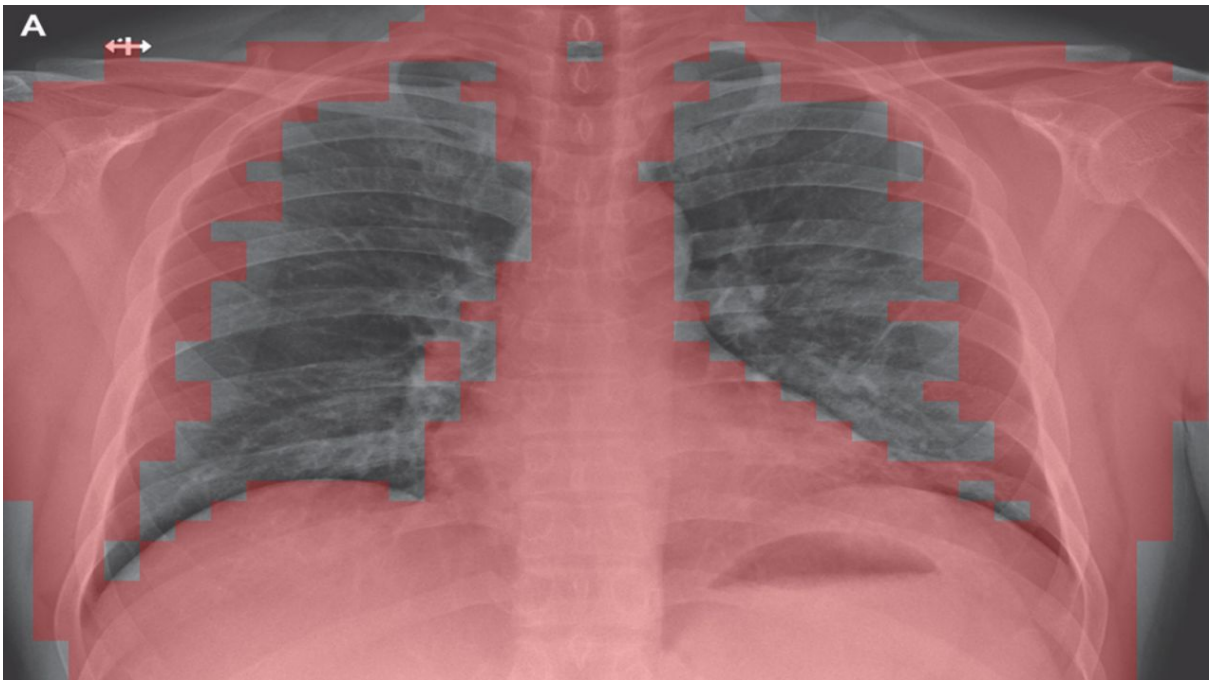

Figura 5.2: Ejemplo de caso de prueba 7

Podemos observar resultados de la clasificación correcta de la neumonía donde como se muestra en la figura 5.1 y la figura 5.2 podemos ver las áreas no marcadas en rojo qué regiones donde se encuentran alveolos infectados.

El problema es que podemos observar que algunas regiones son mal clasificadas debido a la cercanía al tejido del corazón, o incluso a tejido óseo. El problema parece radicar en el número de clases tomadas y la variedad en el estado de las imágenes tomadas.

Para esto quizas lo mas apropiado seria tomar más de una clase, o diferentes datasets que clasifiquen objetos de modo que podamos clasificar el tejido óseo o del corazón tiñéndose de un color y las diferentes neumonías con diferentes colores, de esta manera podríamos tener diferentes configuraciones de clasificación y procesaremos la imagen múltiples veces para clasificar e identificar los diferentes elementos en vez de enfrentar lo que es neumonia vs lo que no es neumonía, pues hay demasiado disparidad en lo que corresponde a la clase negativa del experimento realizado.

Capítulo 6

Conclusiones y trabajo futuro

RESUMEN: En el siguiente capítulo haremos una síntesis de los resultados y conclusiones que nos ha dejado la elaboración y consecución del proyecto así como posibles vías de desarrollo futuro.

6.1. Conclusión

Después de haber realizado el proyecto con la consecución de todas sus fases correspondientes como lo son la fase de investigación o trabajo previo más el desarrollo, podemos destacar que el ámbito del reconocimiento de objetos en aplicaciones médicas puede ser de gran utilidad para estudiantes que están aprendiendo, y quizás en un futuro no tan lejano será útil para realizar diagnósticos mediante máquinas sin necesidad de un doctor sino sólo de un técnico operador que utilice las herramientas apropiadas.

De esta manera, si una herramienta de esta utilidad como lo es la detección del SARS-CoV-2 hubiese sido implementada de manera profesional y eficiente estaríamos hablando de una agilización en la detección de infectados o posibles infectados pudiendo desatascar en la medida de lo real el sistema sanitario.

Está claro que los test rápidos para detectar el SARS-CoV-2 son mucho más asequibles y veloces, sobre todo cuando la pandemia está en fases avanzadas. Esto es debido a que a diferencia de un escaneo por Rayos X que implica una logística y tecnología, así como un gasto de recursos más elevado, estas pruebas rápidas son más rentables, fáciles de manejar, y no requieren condiciones especiales de almacenamiento ni de transporte de equipo, además de que son accesibles para cualquier persona.

El uso de SVM para abordar el problema del reconocimiento de objetos mediante texturas puede dar grandes resultados pero siempre depende de cómo se está enfocando el problema, en este caso la herramienta desarrollada gira en torno a una clasificación binaria, que nos permite identificar ciertos objetos, sin embargo, a la hora de trabajar con casos de neumonía asociada al SARS-CoV-2 es difícil que el análisis concluyente diferencie entre diferentes neumonías víricas, lo propio sería trabajar con múltiples clases en vez de realizar una clasificación binaria.

6.2. Conclusión

After having carried out the project with the achievement of all its corresponding phases such as the research phase or previous work plus development, we can highlight that the field of object recognition in medical applications can be very useful to automate and facilitate the I work for professionals in the health sector when making a diagnosis.

In this way, if a tool of this utility such as the detection of SARS-CoV-2 had been implemented in a professional and efficient manner, we would be talking about a streamlining in the detection of infected or possible infected, being able to unblock the health system as far as possible.

It is clear that rapid tests to detect SARS-CoV-2 are much more affordable and faster, especially when the pandemic is in advanced stages. This is because, unlike an X-ray scan that involves logistics and technology, as well as a higher cost of resources, these rapid tests are more cost-effective, easy to handle, and do not require special storage or transport conditions. of equipment, in addition to being accessible to anyone.

Using SVM to address the problem of object recognition in images by texture analysis can grant great results but it will always depend of how the problem is approached, in this case the tool was developed with a binary classification in mind, which allows us to identify certain objects, however the problem with SARS-CoV-2 associated pneumonia versus other viral pneumonias, since the affliction is similar in texture, so working with more classes may solve this so it can be further tuned.

6.3 Trabajo futuro

Hemos podido observar que los análisis de texturas pueden realizar clasificaciones de diferente formas, esto dependerá en muchos casos de los datos estadísticos extraídos de la imagen junto con la matriz de co-ocurrencia, los ángulos utilizados y la máquina de soporte de vectores que se utilice.

También conviene definir un número de clases apropiadas para el problema tomando en cuenta los diferentes elementos que podemos encontrar en la imagen, en vez de asociar por el objeto que buscamos versus lo demás, ya que esto puede causar problemas de cara a la clasificación, se pueden dar elementos que no pertenezcan a ninguna clase, o elementos que pertenezcan incluso a dos , podría darse el caso de tejido óseo superpuesto a tejido muscular del corazón por ejemplo.

6.2.1. Sistema de ejecución

Uno de los mayores problemas del sistema propuesto es la velocidad de cálculo, extraer todos los datos de la imagen requiere una carga de trabajo algo pesada lo que puede resultar bastante incómodo a la hora de uso, utilizar sistemas como puede ser el caso de Google Colab[11].

Google Colab podría permitir dejar el paso final de la identificación a la nube, de esta manera se podría ejecutar un dataset con cierta configuración, de modo que al finalizar nos diese todos los resultados de clasificación de las imágenes, sin necesidad de disponer de un sistema con mucha potencia, y sin requerir de un uso de recursos de nuestro propio sistema.

6.2.2. Clasificación con múltiples clases

Como se ha ido comentando a lo largo del documento uno de los mayores problemas a la hora de tener éxito con la clasificación de la neumonía asociada a SARS-CoV-2 parece ser el número de clases, extender el sistema para que permita tomar muestras y trabajar con múltiples clases podría ser bastante útil a la hora de clasificar este problema, o incluso en diferentes problemas, ya que cada problema tiene un número de clases óptimo y la posibilidad de determinarlos permite realizar un acercamiento a la solución más óptima mediante un trabajo de prueba y error.

6.2.3. Detección de enfermedades o anomalías fenotípicas

Una aplicación consistente y de gran utilidad para un futuro no muy lejano, sería la detección de enfermedades o anomalías fenotípicas, con esto nos referimos a la detección de anomalías en el bioma humano como el Síndrome de Down. Dicha enfermedad o tipos de enfermedades visuales a nivel facial o corporal podrían ser analizadas del mismo modo pero a través de una cámara común.

Aquí el abanico de posibilidades y detección temprana de enfermedades diagnosticables a nivel visual es inmenso. Si a través de una cámara cualquiera pudieses detectar anomalías en el individuo y analizar las muestras con una aplicación de aprendizaje automático, sería más que posible una detección e intervención temprana de la enfermedad.

[11] https://colab.research.google.com/

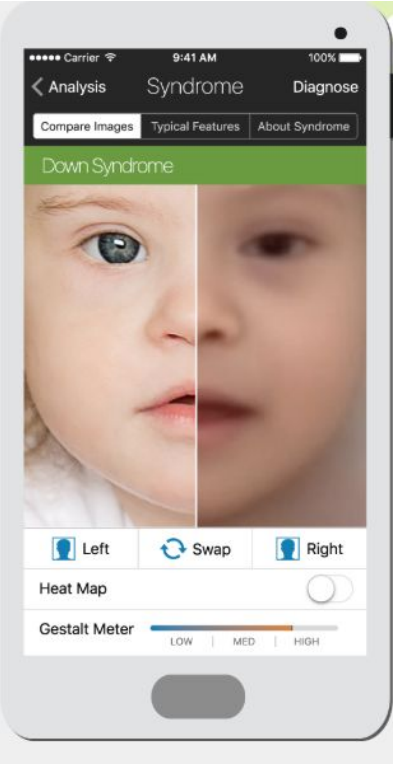

Figura 6.1: Ejemplo de aplicación móvil para la detección del Síndrome de Down

Un ejemplo de aplicación al uso sería Face2Gene, la cual provee de detección de fenotipos desde fotos faciales así como un análisis exhaustivo de la enfermedad a detectar, en este caso en concreto el Síndrome de Down, a través de un DataSet interno con muestras positivas y negativas de la enfermedad.

6.2.4. Aplicación de las SVM para el reconocimiento de en el habla o identificación de voz

A partir de las SVMs podríamos detectar alteraciones en el discurso o identificación de interlocutores para uso terapéutico o forense según el caso. Mediante la obtención de grandes muestras de discurso entre otros patrones en el discurso se podría llegar a un reconocimiento exhaustivo dependiendo del DataSet de muestra dónde se indiquen los casos positivos y negativos en el habla.

En cuanto al uso forense, se podría recabar su uso para la intervención de casos a través de la identificación de sospechosos a través de la intrusión o pinchazo de la línea telefónica y conversación como tal. Sintetizando se podría identificar patrones, ya que las SVMs abordan un problema como lo es la clasificación de patrones de dos clases positivos o aceptados y negativos o rechazados.

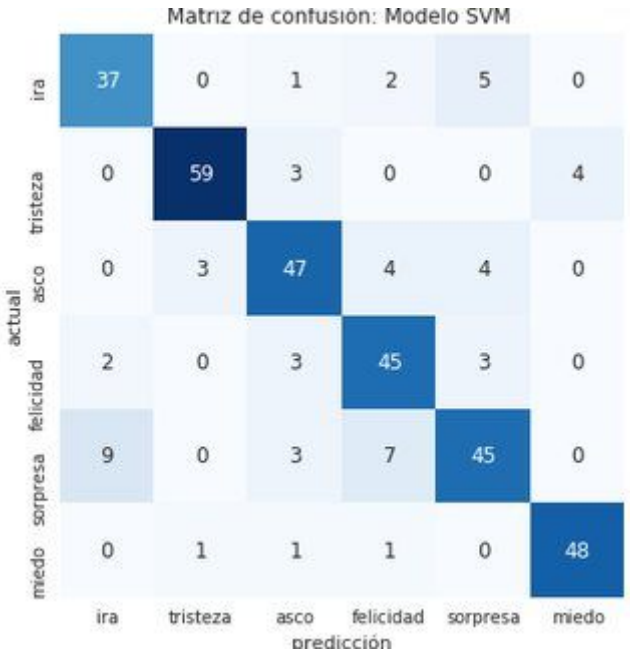

Figura 6.2: Ejemplo de matriz de confusión para el reconocimiento de emociones a través del habla.

Capítulo 7

Contribuciones individuales

RESUMEN: En el siguiente capítulo indicaremos las contribuciones realizadas por los miembros del proyecto

7.1. Joaquín Barrio Lottmann

Al comenzar el proyecto lo dividimos en dos partes, la fase de investigación y la fase de desarrollo. Por mi parte inicialmente me dediqué a buscar herramientas para desarrollo mejor se adaptan al proyecto que buscábamos desarrollar, para ello me apoyé en la documentación inicial entregada por el tutor.

Por otro lado también estuve investigando diferentes temas a los que se les pudiese aplicar la herramienta que estábamos desarrollando.

Inicialmente encontré un dataset interesante que era muestras microscópicas de SARS-CoV-2, al presentar esto como posible opción de análisis se vio que no era realmente algo viable pues se trataban de imágenes de baja calidad y con tamaños muy pequeños de modo que la identificación de objetos no iba a resultar óptima.

Conjuntamente con mi compañero en remoto nos juntamos con cierta regularidad para avanzar en el proyecto, mientras que mi compañero avanzaba un poco más en la bibliografía y yo iba desarrollando lo ya trabajado, siempre en remoto los dos juntos para revisar que todo iba como ambos habíamos comprendido y que no estábamos tomando un camino equivocado.

La comunicación la realizamos siempre vía Skype, de modo que mediante una llamada y compartir la pantalla definimos como avanzar y realizamos el desarrollo. Para esta fase en cierto punto comenzamos a utilizar versiones de control para llevar un registro del avance y para tener un control sobre posibles versiones con peoras.

Finalmente la memoria se distribuye apropiadamente y no fue necesario coordinarse como anteriormente pues libremente cada uno trabajamos en nuestras partes asignadas, y nos reunimos de tiempo en tiempo para revisar los avances entre los dos.

7.2. Xuebo Zhu Chen

Para empezar el proyecto se dividió en dos fases, una fase de investigación y otra de desarrollo. Por lo que a mí respecta, a partir de la bibliografía y los recursos proporcionados por nuestro tutor inicié la puesta en marcha de la fase de investigación sintetizando y buscando material online respecto al tema.

Varios recursos que nos fueron otorgados nos proporcionaron las vías posibles de desarrollo para la elaboración de la herramienta. Habiéndose encargado Joaquín de la parte de desarrollo como Programador, y yo como rol de Analista fuimos construyendo la herramienta conjuntamente de manera remota.

La coordinación y comunicación como se indica en la metodología de trabajo se realizó a través de Skype principalmente. Durante la fase de desarrollo, se hizo un uso leve de herramientas de control de versiones. Siendo el desarrollo conjunto a través de los días los módulos se fueron construyendo secuencialmente. Siempre apoyándonos en las librerías de SVM para Python que ofrecía la comunidad.

Posteriormente, en la realización de la memoria del proyecto la distribución de la tarea también fue dividida. Dicha tarea no necesitaba de tanta coordinación ni comunicación por lo que cada uno fue añadiendo material y contenido a cada uno de los capítulos correspondientes de dicha memoria.

Capítulo 8

Apéndice A: Ejemplo de ejecución

Como ejemplo tomaremos dos imágenes de las muestras usadas para el problema, una para tomar muestras y otra para clasificar con los resultados obtenidos sobre la primera imagen.

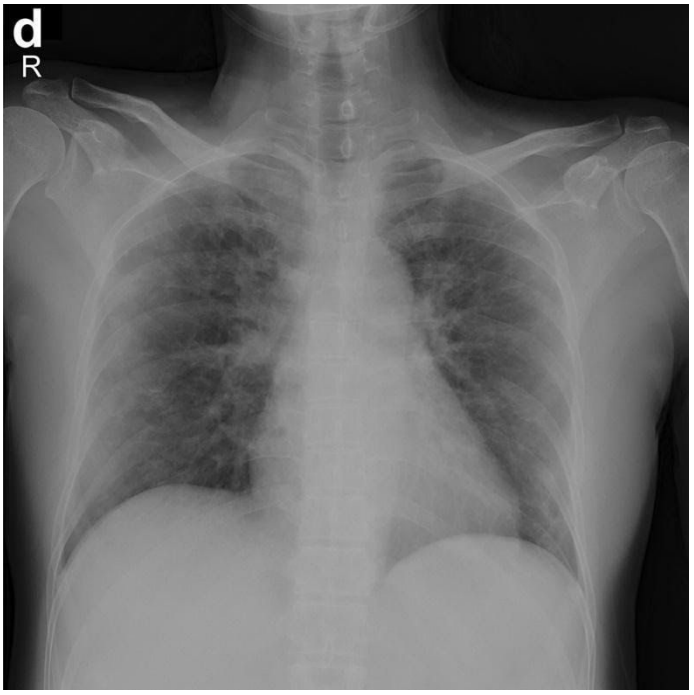

Figura A.1: Ejemplo de imagen virica 1

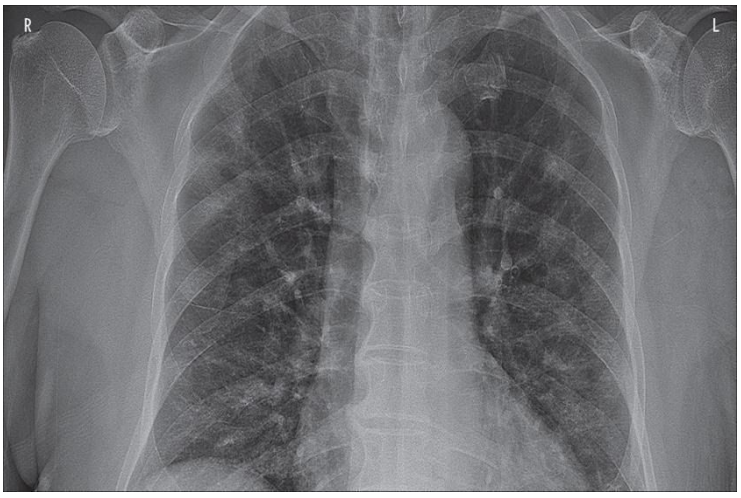

Figura A.2: Ejemplo de imagen vírica 2

Primero mediante el primer módulo de la aplicación tomaremos muestras de la imagen, estas muestras las utilizaremos para formar un modelo de datos con el segundo módulo de la aplicación, con el cual finalmente podremos aplicar a diferentes imágenes con el tercer módulo.

Una vez iniciado el Módulo 1 abrimos una imagen y seleccionamos las características que vamos a utilizar para el estudio.

Para ello nos dirigimos a Archivo -> Abrir, y seleccionamos diferentes características para usar para el estudio que planeamos realizar.

Figura A.3: Ejemplo de imagen vírica 3

Una vez que tenemos las características seleccionadas procederemos pulsando el botón “Seleccionar Características”, procedemos marcando en la imagen muestras

positivas y negativas. Tras indicar las muestras deseadas se pulsa sobre “Procesar Muestras” que tomará los datos acorde a las características seleccionadas, tras esto procedemos a guardar los datos en un zip (Archivo-> Guardar) que será usado en los posteriores módulos.

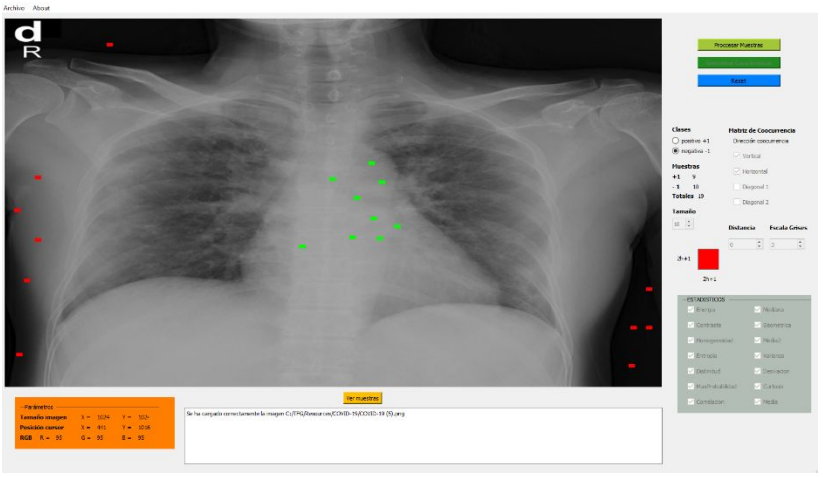

Figura A.4: Ejemplo de imagen vírica analizada

Tras finalizar con la toma de muestras procedemos con el módulo dos para clasificar dichas muestras mediante un SVM.

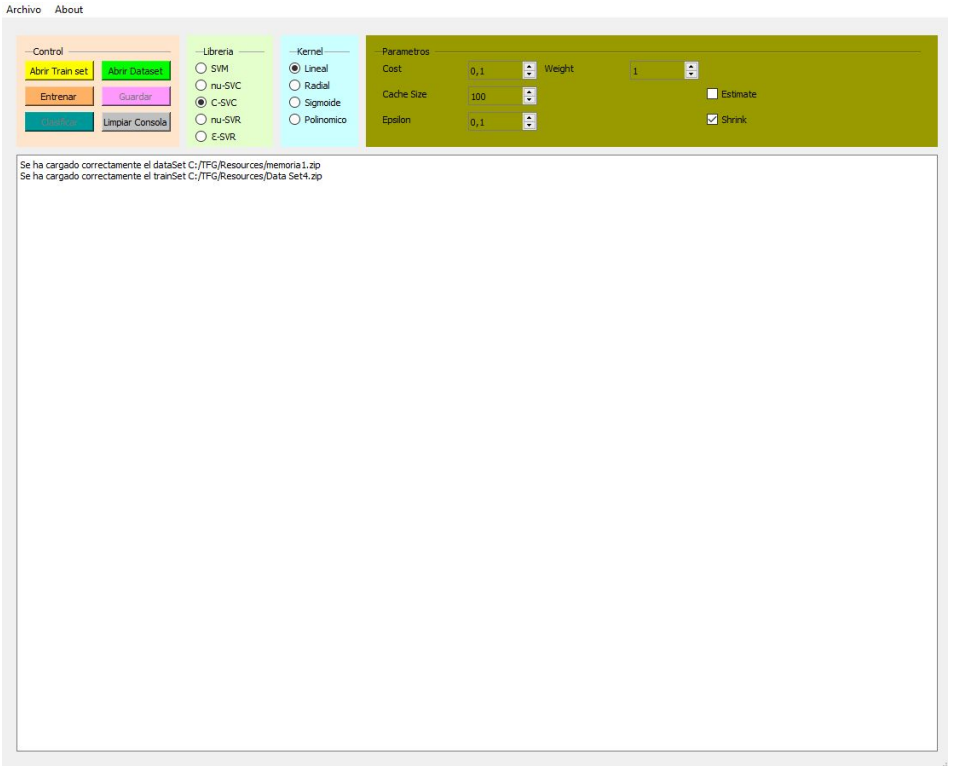

Figura A.5: Ejemplo de utilización de la herramienta

Para proceder abriremos un train set con los datos guardados del zip obtenido del módulo y un dataset que igualmente provenga de muestras tomadas del primer módulo ambos sets de datos los usaremos para trabajar con la SVM.

Una vez que tengamos cargado un train set, podremos entrenar la máquina de modo que seleccionaremos una librería entre las opciones listadas, el kernel a aplicar y modificaremos las características en función del estudio a realizar, tras esto pulsaremos en entrenar.

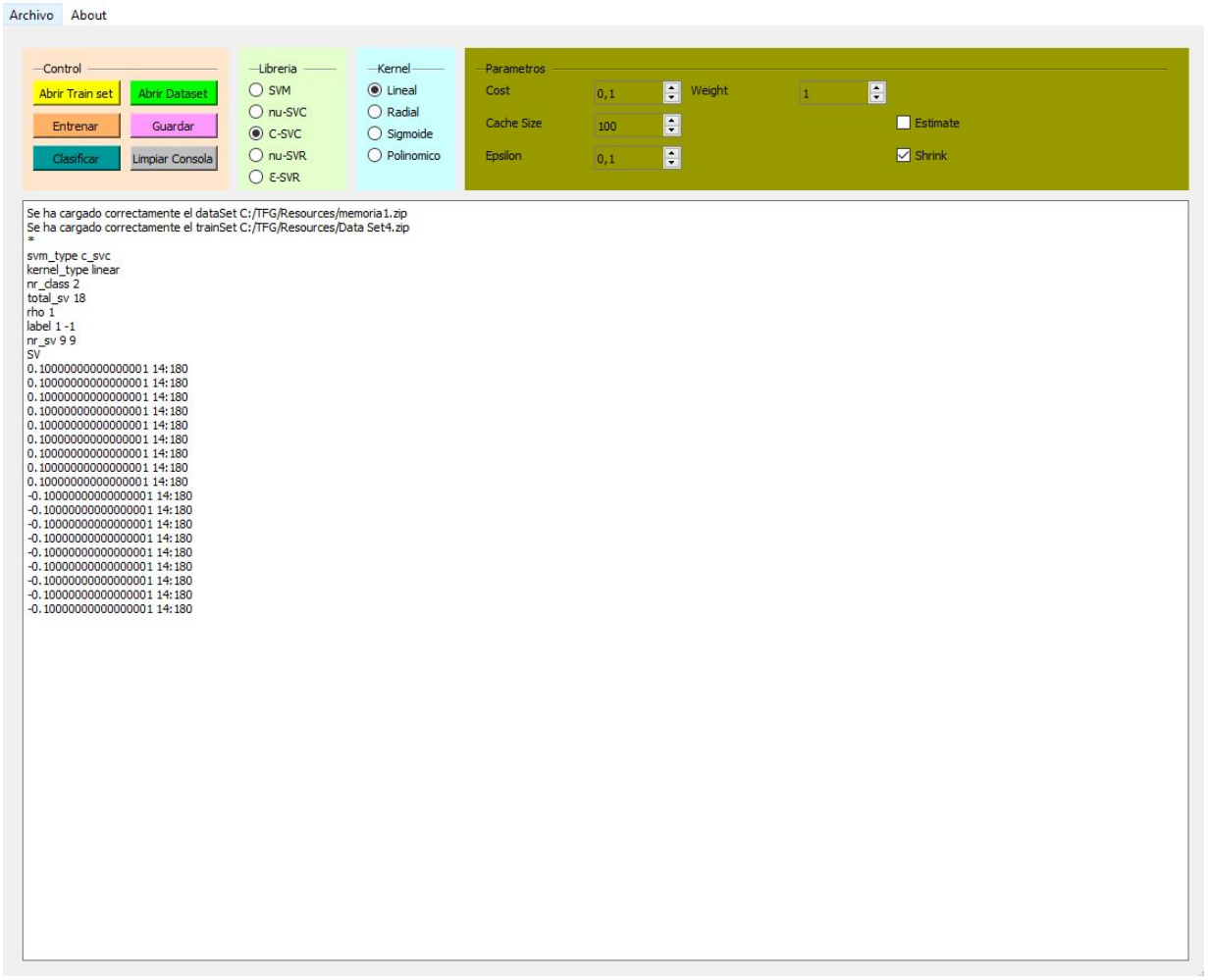

Figura A.6: Ejemplo de uso de la herramienta módulo 2 (Salida por consola).

Una vez entrenada la máquina se habrá generado un modelo, que podremos usar para clasificar otros sets de datos.

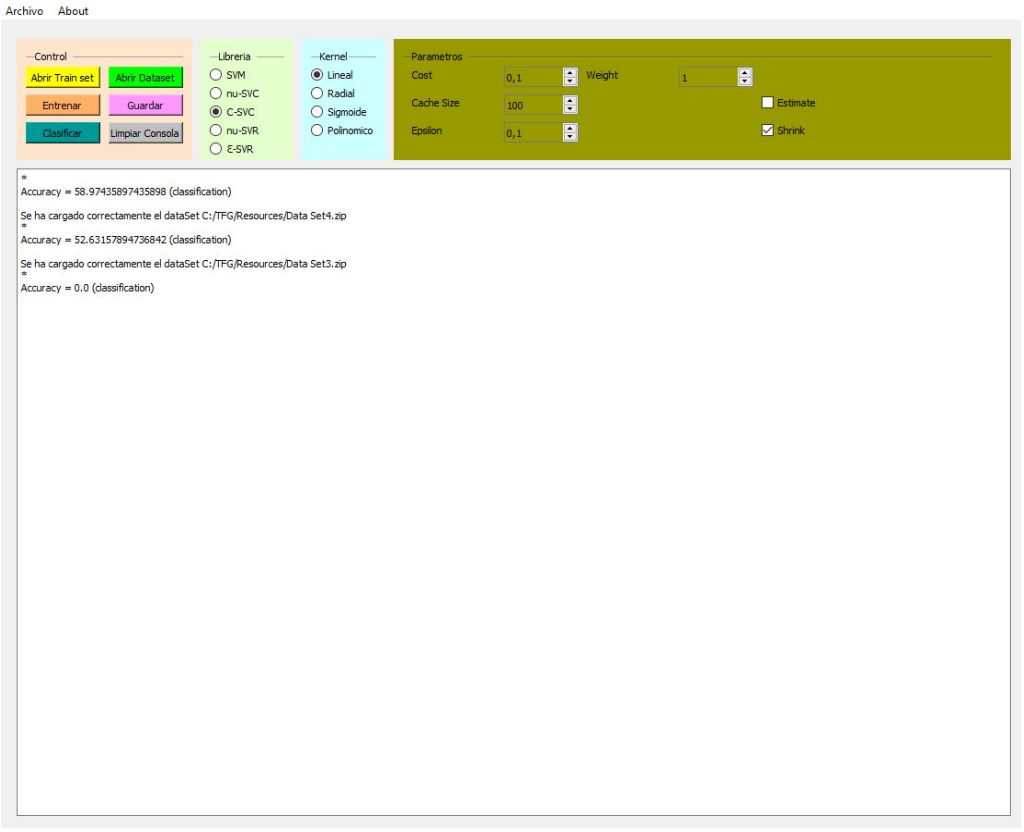

Figura A.7: Ejemplo de uso de la herramienta, ejecución completada

Una vez estamos satisfechos con un modelo y su capacidad de clasificar las muestras del problema, podemos proceder a guardar el modelo y pasaremos al tercer módulo del proyecto

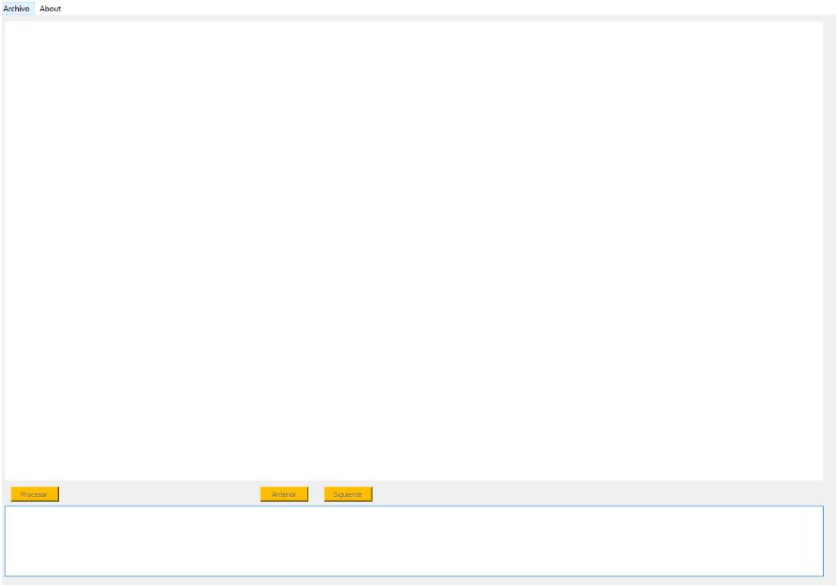

Figura A.8: Ejemplo de uso de la herramienta (Módulo 3)

Para empezar, abriremos una imagen o un grupo de imágenes comprimidas en un zip y un dataset, esto según el modelo de datos podrá tomar un tiempo, ya que lo primero que realizar es entrenar la máquina con el modelo de datas que estamos cargando.

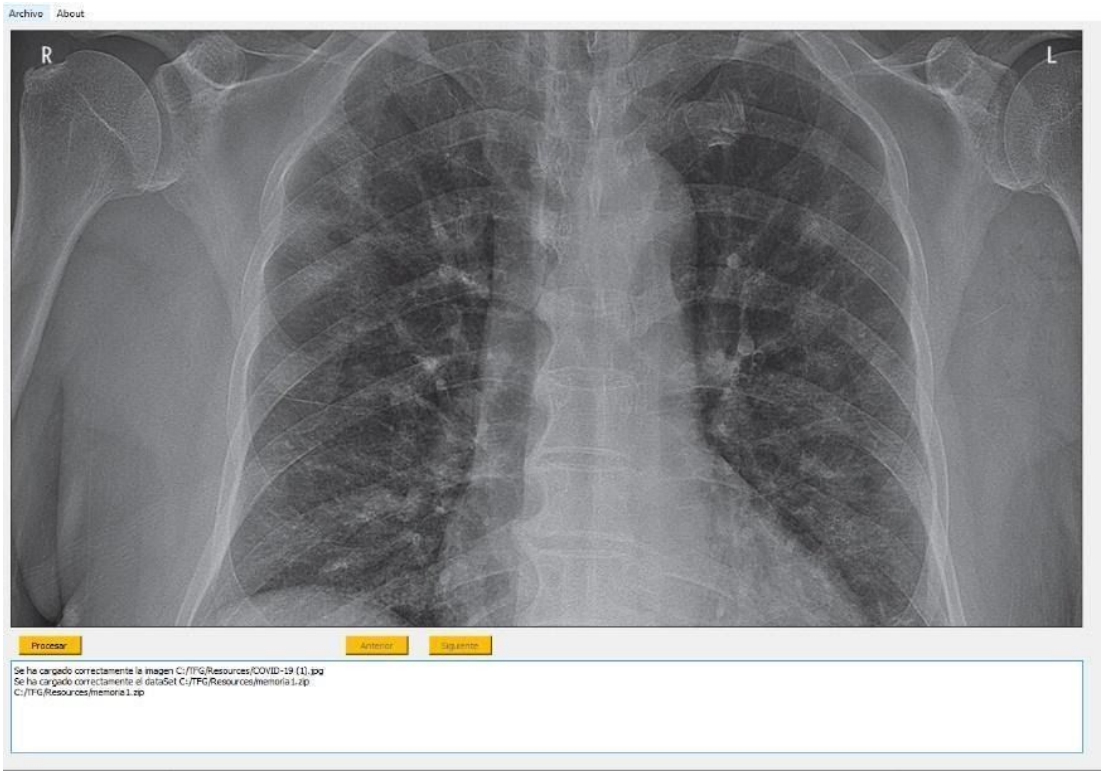

Figura A.9: Ejemplo de uso de la herramienta (Módulo 3, imagen cargada)

Tras cargar el modelo solo nos queda pulsar sobre el botón procesar, que procederá a realizar un barrido tomando todas las muestras de la imagen acorde al DataSet y clasificará mostrándolo visualmente.

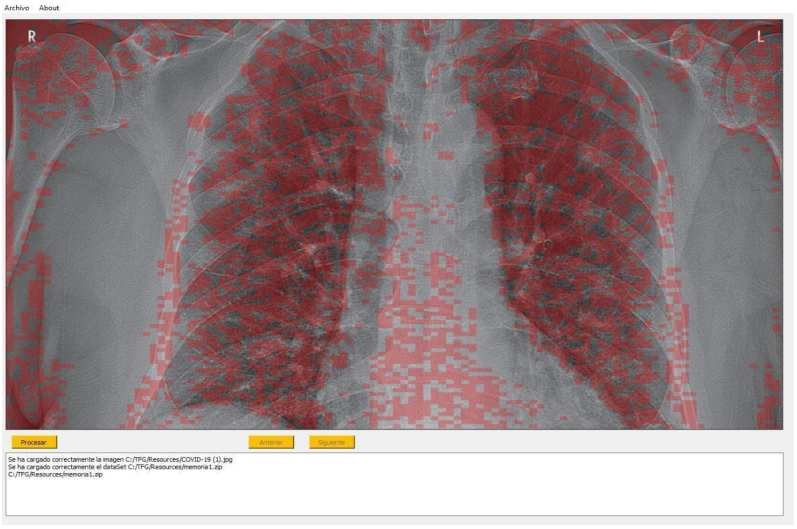

Figura A.10: Ejemplo de uso de la herramienta (Módulo 3, imagen procesada)

Capítulo 9

Apéndice B: Manual de usuario

Para empezar, lo primero es obtener los diferentes módulos, para todos ellos requerimos Python 3.7 acompañado de ciertos módulos extra.

Lo más recomendado para esto es el uso de anaconda [12]que nos permite generar un entorno virtual que se adapta a diferentes entornos y sistemas.

Una vez esté instalado desde la consola de anaconda (conda promt):
conda install -c dsdale24 pyqt5
conda install -c anaconda numpy
conda install -c anaconda pil
conda install -c conda-forge scikit-image

El resto de paquetes necesarios vendrán instalados por defecto en el entorno de Python 3.7 suministrado por anaconda, o formarán parte del código del proyecto

Las siguientes tecnologías y herramientas serán necesarias para la ejecución completa del proyecto

Requisitos:

- Python 3.7
- common (Paquete suministrado en el código)
- Módulo 1 (Paquete suministrado en el código)
- Módulo 2 (Paquete suministrado en el código)
- Módulo 3 (Paquete suministrado en el código)
- Scikit-image
- ZipFile(Python Lib)
- Math(Python Lib)
- IO(Python Lib)

[12] https://www.anaconda.com/

- Threading (Python lib)
- Multiprocessing(Python Lib)
- JSON (Python Lib)
- Pillow
- PyQtT
- Numpy
- Libsvm

Bibliografía

[1] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Introduction to Statistical Learning. 2014.

[2] Alexandre Kowalczyk. Support Vector Machines Succinctly. 2017

[3] Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin. A Practical Guide to Support Vector Classification. 2003

[4] Alexander Statnikov, Douglas Hardin, Isabelle Guyon, Constantin F. Aliferis. A Gentle Introduction to Support Vector Machines in Biomedicine. February 2011. Pages: 200

[5] Ghodselahi, A. A hybrid support vector machine ensemble model for credit scoring. International Journal of Computer Applications 17 (2011).

[6] Hamel, L. H. Knowledge discovery with support vector machines, vol. 3. John Wiley & Sons, 2011.

[7] F. J. M. Navas. Desarrollo de un proceso de deconvolución para imágenes usando GPUs. Facultad de informática, Universidad de Murcia, 2009.

[8] Bellotti, T., and Crook, J. Support vector machines for credit scoring and discovery of significant features. Expert Systems with Applications 36, 2 (2009), 3302-3308.

[9] Berger, S., and Gleisner, F. Emergence of financial intermediaries in electronic markets: The case of online p2p lending. BuR Business Research Journal 2, 1 (2009).

[10] J. A. Rodrigo. Máquinas de Vector Soporte (Support Vector Machines, SVMs). https://rpubs.com/Joaquin_AR/267926, Abril, 2017.

[11] J. L. G. Rodríguez. Estado Actual de la Representación y Análisis de Textura en Imágenes. Julio, 2008.

[12] Cristianini, N., and Shawe-Taylor, J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2000.

[13] Crook, J. N., Edelman, D. B., and Thomas, L. C. Recent developments in consumer credit risk assessment. European Journal of Operational Research 183, 3 (2007), 1447-1465.

[14] de Souza, C. R. Kernel functions for machine learning applications, March 2010.